AFIT/DS/ENS/94-1

# Feature and Model Selection
# in
# Feedforward Neural Networks

DISSERTATION

Jean M. Steppe
Captain, USAF

AFIT/DS/ENS/94-1

Approved for public release; distribution unlimited

AFIT/DS/ENS/94-1

# Feature and Model Selection
# in
# Feedforward Neural Networks

DISSERTATION

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Jean M. Steppe, B.S., M.S.
Captain, USAF

June, 1994

Approved for public release; distribution unlimited

AFIT/DS/ENS/94-1

# Feature Selection
# in
# Feedforward Neural Networks

**Jean M. Steppe, B.S., M.S.**

**Captain, USAF**

**Approved:**

______________________________ ____________

**Lt Col Kenneth W. Bauer, Jr., Chairman** **Date**

______________________________ ____________

**Dr. Steven K. Rogers, Committee Member** **Date**

______________________________ ____________

**Dr. Mark E. Oxley, Committee Member** **Date**

______________________________ ____________

**Capt Dennis W. Ruck, Committee Member** **Date**

**Accepted:**

______________________________ ____________

**Dr. J. S. Przemieniecki, Senior Dean** **Date**

## *Acknowledgments*

My gratitude goes to my advisor and committee chairman Lt Col Kenneth W. Bauer, Jr. for the numerous simulating conversations we had and for his constant support and enthusiasm throughout the realization of this dissertation. I also want to thank my other committee members, Dr. Steve Rogers, Dr. Mark Oxley, Captain Dennis Ruck, and the Dean's representative Professor Daniel Reynolds, for their specific comments and suggestions after reading this dissertation.

To my colleagues, Dr. Mark Gallagher, Lisa Belue, Dan Zalewski, Ken Fielding, and Dennis Benson, thank you. The time we all spent talking about topics such as gradients, covariance matrices, and likelihood ratios, to name a few, was an important factor in maintaining steady research progress. A special thanks goes to Lisa Belue for her valuable feedback and insights throughout my research. Another source of important support came from Nancy Freese, who kept all of our computers in commission. On the subject of computer support, I would be remiss if I didn't also acknowledge the countless times that Dan Zalewski, the ultimate source of computer wizardry, helped me to conquer my computer related difficulties. I would also like to acknowledge Jacqueline Logan who over the years has helped me on administrative details more times than I can count.

My heartfelt appreciation goes to my confidants and all-weather running partners, Mark Gallagher, Rich Bagnell, Renee Ingham, and Rich McEachin, whose day-to-day feedback facilitated keeping the 'big-picture' in perspective. Finally, the support of my family must be recognized. I would like to thank my husband Robert, and my children, Holynne, Patrick, and Siobhan for their encouragement and for giving me the freedom to spend the numerous week nights and weekends at school over the past three years.

Jean M. Steppe

## *Table of Contents*

Page

Page

Page

## *List of Figures*

## *List of Tables*

AFIT/DS/ENS/94-1

## *Abstract*

This research advances feature and model selection for feedforward neural networks. Feature selection involves determining a good feature subset given a set of candidate features. Model selection involves determining an appropriate architecture (number of middle nodes) for the neural network. Specific advances are made in: neural network feature saliency metrics used for evaluating or ranking features, statistical identification of irrelevant/noisy features, and statistical investigation of reduced neural network architectures and reduced feature subsets. Additionally, a comprehensive statistically-based methodology is presented for feature and model selection.

New feature saliency metrics are presented which provide a more succinct quantitative measure of a feature's importance than other similar metrics. A catalogue of feature saliency metric definitions and interrelationships is developed which consolidates the set of available metrics for the neural network practitioner.

A statistical screening procedure for identifying noisy features is presented. The procedure involves statistically comparing the saliency of candidate features with the saliency of a known noisy feature. Noisy features are successfully identified over a series of test problems using the new saliency screening procedure.

Two novel neural network selection algorithms are developed by posing the neural network model as a nonlinear regression statistical model. The first is an architecture selection algorithm and the second is a feature selection algorithm. The feature selection algorithm is unique because architecture reduction is investigated as features are removed. Both algorithms use the likelihood ratio test statistic within a backwards sequential procedure. Application results demonstrate how these algorithms can be used to search for a more parsimonious neural network model with equivalent prediction accuracy.

A comprehensive neural network selection methodology is developed for identifying both a good feature set and an appropriate neural network architecture for a specific situation. It encompasses a combination of the statistical screening and the statistical architecture and feature selection procedures. Application results demonstrate the utility of the methodology.

# Feature and Model Selection

# in

# Feedforward Neural Networks

## *I. Introduction*

This research advances feature and model selection for feedforward neural networks. Feature selection involves determining a good feature subset from a set of candidate features, and model selection involves determining an appropriate neural network architecture (number of middle nodes).

Generally speaking, feedforward neural networks are used as either regression functions or discriminant functions. For both linear and nonlinear regression analysis, as well as discriminant analysis, there is statistical theory formalizing the process of feature and model selection. Until this research, statistical theory has not been practically used to formalize the selection process for feedforward neural networks.

These research results contribute to theory formalizing feature and model selection for feedforward neural networks. Specific advances are made in: neural network feature saliency metrics used for evaluating or ranking features, statistical identification of irrelevant/noisy features, and statistical investigation of reduced neural network architectures and reduced feature subsets. Additionally, a comprehensive statistically-based methodology is presented for feature and model selection. The remainder of this chapter provides background on feature selection and feedforward neural networks, and a preview of the dissertation.

### *1.1 Feature Selection*

Feature selection as considered in this research involves determining a subset of candidate features specifically in the context of estimating a sufficiently accurate neural network prediction function. Figure 1 shows how feature selection fits into an overall prediction process. In this section, the term 'prediction function' is general and is also used to refer to classification functions.

There are many reasons for using feature selection techniques to reduce the number of features. Reasons for using feature selection techniques include:

- satisfying the general goals of maximizing the accuracy of the prediction function while minimizing the associated measurement costs
- improving prediction accuracy by reducing irrelevant and possibly redundant features
- reducing the complexity and the associated computational costs of a prediction function
- reduce the amount of data needed for accurate prediction (i.e. reduce the 'curse of dimensionality' [14:487]).
- reducing associated data collection and data processing cost
- improving the chances that a solution will be both understandable and practical
- improving the possibility of graphical representation of the data

Generally, a prediction function is finely tuned to the finite amount of available training data. When "feature spaces" are plagued with *irrelevant* or *redundant* features, a prediction function may not generalize well for predicting unknown data, particularly if there is insufficient training data. A reduction in the number of features may degrade prediction accuracy on the training data, but it also reduces the amount of data required for good generalization. Foley recommends the ratio of training vectors in a class to the dimensionality of the feature space should be greater than three to ensure that the error rate on held out data is close to the true error rate [17:623].

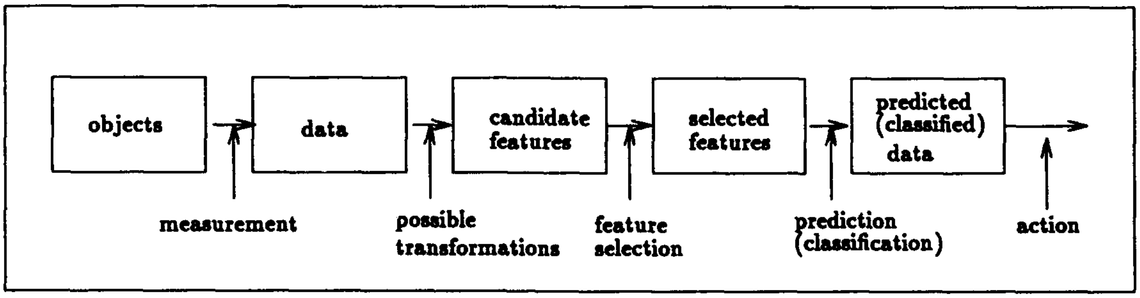


Figure 1. Prediction Process for Regression and Discrimination Problems

His results are based on empirical results for a two-class discrimination problem with multivariate normal distributions for the input features.

In this research a formalized feature selection process is characterized by three components. The first component is a metric or criterion function for evaluating and ranking the features (or feature subsets). The second component is a set of screening procedures for identifying irrelevant and redundant features. The third component is a search methodology for examining possible feature subsets. The results of this research provide advances to all three components of the feature selection process in the context of feedforward neural networks.

## *1.2 Feedforward Neural Networks*

Feedforward networks are generally used in two types of applicatiorns: regression analysis and discriminant analysis. For regression applications, the network is used ιo estimate a linear or nonlinear function for prediction. For discriminant analysis applications, the network is used to estimate a linear or nonlinear discriminant function for classification. Covered in the remainder of this section are:

- an overview of feedforward neural networks
- the backpropagation algorithm
- the neural network approximation to the Bayesian optimal discriminant function
- confidence interval estimation techniques

*1.2.1 Feedforward Neural Networks Overview.* Feedforward neural networks, often referred to as multilayer perceptrons, generally have a feature input layer, one or more hidden layers, and a function output layer. The neural network shown in Figure 2 illustrates the structure and notation associated with a single hidden layer feedforward neural network. The notation will be defined in Section 1.2.2.

The feature input layer consists of normalized feature input data. The feature input data can be the raw data or an appropriate 'ransformation (or projection) of the raw data. Typical normalization of the feature inputs consists of either a simple transformation so that all features

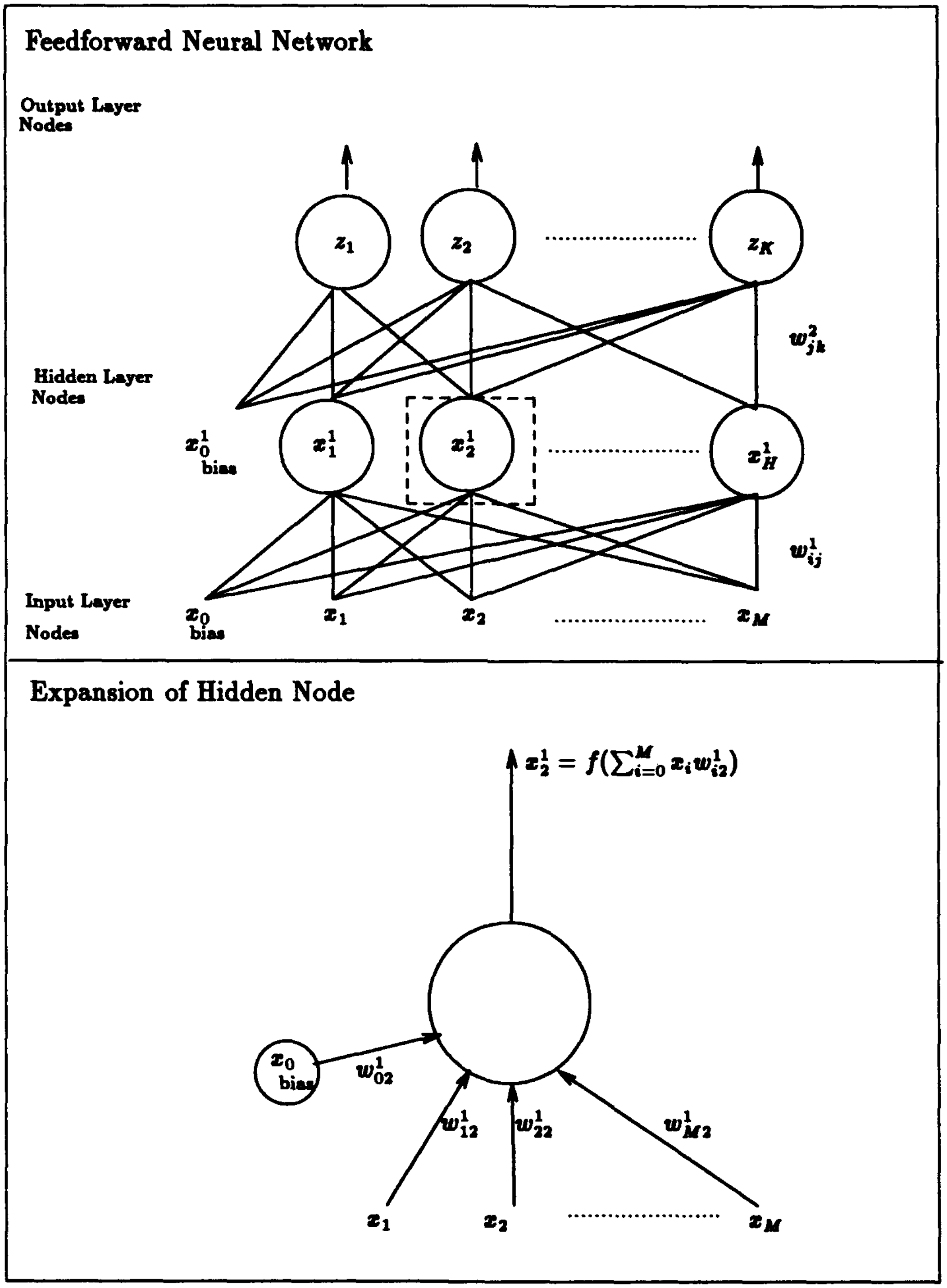


Figure 2. Single Hidden Layer Feedforward Neural Network

have the same range, say between $-1$ and 1, or between 0 and 1, or a statistical normalization which standardizes each feature to null mean and unit variance [41:50] [60:16] [76:100].

Depending on the application, the nodes on the hidden (middle) layers either have linear or nonlinear activation functions $f(a)$. For example, linear activation functions, where $f(a) = a$, could be used in linear regression. The sigmoidal activation function is commonly used as a nonlinear activation function since its derivative is continuous and makes the weight update rule simple for backpropagation training. The sigmoidal activation function and its derivative are defined as:

$$\begin{aligned} f(a) &= \frac{1}{1+e^{-a}} \\ \frac{\partial(f(a))}{\partial x} &= f(a)[1-f(a)]\frac{\partial a}{\partial x} \end{aligned}$$

A single hidden layer network with sigmoidal squashing functions on the hidden layer is used in this research. A single hidden layer configuration is common because Cybenko, and Hornik and others', show that this type of network (with linear output nodes) is capable of arbitrarily accurate approximations for any arbitrary function provided a sufficient number of hidden nodes are used with either sigmoidal activation functions [12], or appropriately smooth activation functions [27]. Although the number of required hidden nodes is unknown in advance, a reasonable number of middle nodes is often determined by a trial and error process or by more sophisticated methods [9, 13, 25, 28, 31, 32, 37, 44, 52, 64, 69, 79, 83]. A by-product of the feature selection research done in this dissertation is a novel architecture selection algorithm presented in Chapter V for investigating the appropriate number of middle nodes. The algorithm is unique because it is based on a nonlinear statistical model selection criterion.

The function output layer consists of one or more nodes with linear or nonlinear activation functions depending on the application. Linear output activations are generally used for function approximation. Nonlinear sigmoidal output activation functions are generally used for discriminant function applications. In this research, the output layer has sigmoid activation functions since discrimination function applications are used.

*1.2.2 Backpropagation Training Algorithm.* Backpropagation is the most popular algorithm for finding a feedforward neural network's weight parameters. The backpropagation algorithm was first developed by Werbos [77]. Later, it was rediscovered independently by Parker and then reformulated by Rumelhart, Hinton and Williams with reference to prior work by Parker[48, 63]. A good overview of the backpropagation training algorithm is in Lippmann [40, 41].

Backpropagation is an iterative gradient descent algorithm requiring sample problem data. It involves minimizing the error between the actual and desired outputs of the network in order to estimate a neural network's optimal weight parameters. In the backpropagation algorithm described herein, the weights are "instantaneously" updated after the presentation of each input vector. In another version of backpropagation, batch backpropagation, the weights are only updated after the error gradient has been aggregated for one full presentation or "epoch" of the training data. The instantaneous back propagation algorithm followed by additional details is presented next.

### The Instantaneous Backpropagation Algorithm<br>for a<br>Single Hidden Layer Feedforward Neural Network

1. Randomly partition data into *training*, *training-test*, and *validation* sets.
2. Normalize the feature input data.
3. Initialize weights to small random values.
4. Present the network with a randomly selected vector from the training set, denoted $\mathbf{x}^p$.
5. Calculate the network output $\mathbf{z}^p$ associated with the $p$th training vector.
    - $k$th neural network output: $z_k^p = f(\sum_{j=0}^{H} w_{jk}^2 x_j^1)$, where
        - $H$ is the number of middle nodes
        - $f(a) = 1/(1 + e^{-a})$ for sigmoidal activation functions
        - $f(a) = a$ for linear activation functions
        - $w_{jk}^2$ is the weight from middle node $j$ to output node $k$
        - $x_0^1$ is the middle layer bias term and is set equal to 1
        - $x_j^1 = f(\sum_{i=0}^{M} w_{ij}^1 x_i^p)$ is the output of middle node $j$
        - $M$ is the number of feature inputs
        - $w_{ij}^1$ is the weight from input node $i$ to middle node $j$
        - $x_0^p$ is the input layer bias term, and is equal to 1
        - $x_i^p$ is the $i$th feature input
6. Update the weights.
    - upper layer weights: $(w_{jk}^2)^+ = (w_{jk}^2)^- + \eta\delta_k^2 x_j^1$,

- lower layer weights: $(w^1_{ij})^+ = (w^1_{ij})^- + \eta\delta^1_j x^p_i$, where
  - $(w^2_{jk})^+$ is the updated weight from middle node $j$ to output $k$
  - $(w^2_{jk})^-$ is the old weight from from middle node $j$ to output $k$
  - $(w^1_{ij})^+$ is the updated weight from input $i$ to middle node $j$
  - $(w^1_{ij})^-$ is the old weight from from input $i$ to middle node $j$
  - $\eta$ is the step size
  - $\delta^2_k = (d^p_k - z^p_k)z^p_k(1 - z^p_k)$ if there is a sigmoid on the output
  - $\delta^2_k = (d^p_k - z^p_k)$ if the output is linear
  - $\delta^1_j = x^1_j(1 - x^1_j)\sum_{k=1}^{K}\delta^2_k(w^2_{jk})^-$, if there is a sigmoid on middle node $j$
  - $\delta^1_j = \sum_{k=1}^{K}\delta^2_k(w^2_{jk})^-$, if middle node $j$ is linear
  - $d^p_k$ is the $k$th desired output of the $p$th exemplar

7. If training-test set error does not indicate sufficient convergence, go to step 4.

In Step 1, the problem data is randomly divided into two sets: a *training* data set and a *validation* data set [23:116-117]. The *training* data set is further subdivided into a *training* set and a *training-test* set. The *training* set is used to estimate the weight parameters, and the *training-test* set is used to evaluate the backpropagation learning process by measuring the network's performance on unknown data. The *validation* data set is a set of data which has not been used in any way to determine the prediction function. The *validation* data set is used to evaluate a neural network's capability to adequately generalize to future data. Some guidelines on determining these data sets are given in [23:116-119] [76:28-39].

Data normalization schemes which can be used for feature input normalization in Step 2 are discussed in Section 1.2.1. In this research, the validation data set is normalized separately from the *training* data. This keeps the normalization information for the *training* and *validation* data sets separate.

In Step 3, weight parameters are usually initialized to small random numbers between −.5 and .5 [55:56]. Step 4 of the backpropagation algorithm is characterized by feeding a randomly selected feature input vector $\mathbf{x}^p$ into the neural network, where $p$ indicates that $\mathbf{x}$ is the $p$th vector in the training set. Step 5 involves calculating the vector of network outputs in a feedforward manner via the summations and sigmoids defined by the network's structure.

In Step 6, the instantaneous network output error $\mathcal{E}_o^p$ associated with $\mathbf{x}^p$ is calculated using the $p$th vector of neural network outputs $\mathbf{z}^p$ and the corresponding vector of desired outputs $\mathbf{d}^p$. Instantaneous network output error $\mathcal{E}_o^p$ is the squared error associated with the $p$th exemplar and is given as:

$$\mathcal{E}_o^p = \sum_{k=1}^{K}(d_k^p - z_k^p)^2 \qquad (1)$$

where $K$ is the number of output nodes, $d_k^p$ is the desired output associated with the $p$th exemplar and $k$th output, and $z_k^p$ is the network output with the $p$th exemplar and the $k$th output. The gradient descent step direction is determined by taking the partial derivative of $\mathcal{E}_o^p$ with respect to the weight parameters. A derivation of the gradient descent step direction used in Step 6 is given by Rogers and others [55].

The step size, $\eta$, can be constant or variable. White makes the point that a constant learning rate is inefficient because the random influences in the input will result in random fluctuations in the weight vector preventing backpropagation from ever settling down to the optimal weight vector [80]. A declining learning rate (eventually declining to zero) is minimally required for backpropagation to settle down [80]. White suggests declining learning rates which are inversely proportional to the number of epochs or the log of the number of epochs [80]. Three potential declining learning rates are defined below in terms of: the total number of epochs $N_e$, the current epoch $L$, and the starting value $a$ for a linearly declining learning rate. The drawback of the linearly declining learning rates is that the maximum number of training epochs is used, which is not generally known in advance.

$$\text{Log Declining Rate } \nu_L = \frac{1}{ln(1+L)}$$

$$\text{Linearly Declining Rate } \nu_L = a\left(1-\frac{L}{N_e+1}\right)$$

$$\text{Log-Linearly Declining Rate } \nu_L = \frac{\left(1 - \frac{L}{N_e+1}\right)}{ln(1+L)}$$

Two types of error rates are associated with backpropagation training: output error and classification error. Output error is measured as a function of the approximation error between the vector of network outputs $\mathbf{z}$ and the vector of desired or true outputs $\mathbf{d}$. Typically, the output error is measured as the average squared network error. For a training set of $P$ vectors, the output error, denoted $\mathcal{E}_o$, is defined

$$\mathcal{E}_o = P^{-1}\sum_{p=1}^{P}\sum_{k=1}^{K}(d_k^p - z_k^p)^2, \tag{2}$$

where $P$ is the number of exemplars in the training set, $K$ is the number of output nodes, $d_k^p$ is the desired output associated with the $p$th exemplar and the $k$th output, and $z_k^p$ is the network output with the $p$th exemplar and the $k$th output.

Classification error is used to measure the percentage of vectors which are incorrectly classified in a data set. This type of error is applicable only when neural networks are used for discriminant analysis or pattern classification problems. Define $I^p$, a Bernoulli random variable as:

$$I^p = \left\{ \begin{array}{ll} 1 & \text{if } x^p \text{ is incorrectly classified} \\ 0 & \text{otherwise} \end{array} \right\}.$$

where $x^p$ is the $p$th exemplar. The classification error, denoted $\mathcal{E}_c$, is then defined as the average of $P$ Bernoulli random variables $I^p$:

$$\mathcal{E}_c = P^{-1}\sum_{p=1}^{P} I^p$$

The random partitioning of the data in Step 1 is often referred to as the *hold-out* method when discussed in the context of error estimation. The hold-out method gives a conservative estimate of the average network error for two reasons. One, it does not use all available data while training the classifier [14:355]. Two, it uses an independent validation data set, rather than the training-test set, to measure the average network error.

Minimum output error (and minimum classification error if appropriate) on the *training-test* set is a good indicator of sufficient convergence or good 'generalization capability.' In Step 7, the algorithm terminates if the error rate has converged sufficiently or the algorithm continues back to Step 4 to get a new training vector.

Over-training is an undesirable phenomena which sometimes occurs with backpropagation training. This phenomena has occurred if the *training-test* set error begins to increase while the *training* set error continues to decrease. Over-training indicates the *training* set has been memorized at the expense of the network's capability to predict (generalize to) the *training-test* set. In this research, over-training was observed in cases of small training sets where a neural network with more than enough middle nodes were used. The combination of a small training data set with a large number of middle nodes gives the network the capability to memorize or to over-generalize to the training set.

### *1.2.3 Approximation to the Bayesian Optimal Discriminant Function.*

In this section, the neural network approximation to the Bayes optimal discriminant is discussed. The Bayes optimal discriminant function minimizes the probability of error. It can be defined for the $k$th class of a multi-class problem using the posterior probability of $\mathbf{x}$ belonging to class $k$ [61]. The vector $\mathbf{x}$ is classified as belonging to class $k$ if the largest discriminant function value is from the $k$th class.

Several researchers have proven that a neural network approximates a Bayes optimal discriminant under certain conditions [30, 54, 61, 68, 75] [41:50]. These conditions are:

- The neural network is trained to outputs of 0 and 1.
- The training set data are random variables.
- The network is trained to a minimum mean square error measure.

- The training set class membership percentages reflect the real world.

When these conditions are met the neural network outputs $z_k$ can be interpreted as approximations to the posterior probability for class $k$. The quality of these approximations is affected by:

- Neural network complexity (i.e. number of middle nodes)
- Amount of training data
- Convergence of the neural network to a solution

*1.2.4 Confidence Interval Estimation.* Confidence intervals can be used to assess neural network error rates. The performance over several runs of a neural network can be characterized as an average error rate. A confidence interval for the average error provides information as to the point statistics variability. Generally, more observations on a random variable will reduce the corresponding confidence interval length.

The error rate of a neural network can be considered an independent random variable dependent on: the random order of the training data set, the random starting point used to determine the weight parameters, and the selected termination point of the neural network training. For reasonably large number of training vectors $P$, the standard normal distribution or the $t$ distribution can be used for confidence interval estimation, depending on whether the variance is known or must be estimated. These distributions are appropriate for large $P$, since the error is independent and approximately normally distributed by the central limit theorem [47:6]. For small data sets, confidence intervals for proportions may also be appropriate for classification error rates [24:252-254].

Let $\mathcal{E}_i, i = 1, \cdots, N$ be a random sample of $N$ observations of the neural network error rate $\mathcal{E}$ which are assumed to be normally distributed. The mean and variance of $\mathcal{E}$ are defined as:

$$\mathrm{E}\{\mathcal{E}\} = \mu$$

$$\mathrm{var}\{\mathcal{E}\} = \sigma^2$$

where E$\{\cdot\}$ is the expectation operator and var$\{\cdot\}$ is the variance operator. The corresponding unbiased and consistent estimators for E$\{\mathcal{E}\}$ and var$\{\mathcal{E}\}$ are $\bar{\mathcal{E}}$ and $s^2$, respectively, which are defined

$$\begin{aligned}\bar{\mathcal{E}} &= N^{-1}\sum_{i=1}^{N}\mathcal{E}_i \\ s^2 &= (N-1)^{-1}\sum_{i=1}^{N}(\mathcal{E}_i - \bar{\mathcal{E}})^2\end{aligned}$$

The mean error $\bar{\mathcal{E}}$ is also approximately normally distributed with an expected value of $\mu$ and variance of $\frac{\sigma^2}{N}$ [47:7]. That is:

$$\begin{aligned}\mathrm{E}\{\bar{\mathcal{E}}\} &= \mu \\ \mathrm{var}\{\bar{\mathcal{E}}\} &= \frac{\sigma^2}{N}\end{aligned}$$

The corresponding unbiased and consistent estimators for E$\{\bar{\mathcal{E}}\}$ and var$\{\bar{\mathcal{E}}\}$ are $\bar{\mathcal{E}}$ and $\frac{s^2}{N}$, respectively.

When $\sigma^2$ is known, the standard normal distribution can be used to form confidence intervals for the expected value of $\bar{\mathcal{E}}$. In practice, however, the true variance is unknown and must be estimated with $s^2$. Therefore, the $t$ distribution is most appropriate for forming confidence intervals for E$\{\bar{\mathcal{E}}\}$ with the statistic

$$\frac{\bar{\mathcal{E}} - \mu}{\frac{s}{\sqrt{N}}}$$

which is distributed as a $t$ distribution with $N-1$ degrees of freedom. Confidence intervals for E$\{\bar{\mathcal{E}}\}$ using the $t$ distribution look like:

$$\bar{\mathcal{E}} - \frac{s}{\sqrt{N}}t_{(1-\frac{\alpha}{2};N-1)} < \mu < \bar{\mathcal{E}} + \frac{s}{\sqrt{N}}t_{(1-\frac{\alpha}{2};N-1)}$$

where $t_{\left(1-\frac{\alpha}{2};N-1\right)}$ is determined by the $t$ distribution for a given confidence coefficient $1-\alpha$ and degrees of freedom $N-1$. One can be $100(1-\alpha)$ percent confident that the absolute error in estimating the $\mathrm{E}\{\mathcal{E}\}$ is less than the confidence interval half width of $\frac{s}{\sqrt{N}}t_{\left(1-\frac{\alpha}{2};N-1\right)}$.

When computing confidence intervals for neural networks, one factor to consider is that backpropagation learning may or may not converge to a local minima [80:143]. Usually, one does not want to corrupt neural network point statistics and confidence intervals by including a network which has not converged. According to White, it makes sense to train a number of neural networks and select the network which minimizes network error [80:143]. Although this type of methodology does not guarantee being close to a global minima, it usually yields estimated network parameters which are consistent for a local minima [80:143].

In this research, the inconsistency of backpropagation training is taken into consideration. An attempt is made to only use neural network results which correspond to networks which are trained to *good* local minima. For a network to be considered 'trained,' the network is required to attain a predetermined (for the problem at hand) maximum error rate on the training set. When a reasonable error rate is not attained, it is assumed that the network has not converged to a good local minimum, and the results from this network are not used. In some cases, it is impractical to enforce a reasonable error rate. In these cases, only a subset of the neural networks are used to compute the statistics for network error. Each network in the subset represents the best network from a sub-experiment where only the best network is kept from a number of trained neural networks.

### *1.3 Preview*

The remainder of this dissertation is organized as follows: Chapter II provides background on the feature selection techniques associated with regression, discriminant analysis, and neural networks. In Chapter III, novel feature metrics for evaluating and ranking candidate features are defined and evaluated, and theoretical relationships among the set of available feature metrics are documented. A technique for statistically identifying noisy features is presented in Chapter IV. In Chapter V, selection algorithms are developed using a nonlinear regression statistical model building perspective for both architecture determination and feature input selection in neural networks. Then, in Chapter VI, a comprehensive neural network selection methodology is developed for identifying both a good feature set and an appropriate neural network architecture for a specific situation. The research is summarized and future research recommendations are made in Chapter VII.

# *II. Background on Feature Selection Techniques*

## *2.1 Introduction*

This chapter provides a complete review of feature selection techniques developed for regression, discriminant analysis, and neural networks. The feature selection techniques developed for regression and discriminant analysis, as well as those developed for neural networks, are relevant background material since neural network applications include problems which have often been solved using more classical regression and discriminant analysis techniques. Feature selection techniques formally developed for classical regression and discriminant analysis may also be potentially useful in a neural network context.

In Section 2, some common feature (variable) selection criteria and methodologies for linear regression are covered. Statistical variable selection criteria for univariate nonlinear regression are covered in Section 3. In Section 4, feature evaluation criteria and selection methodologies are reviewed for discriminant analysis. The feature evaluation metrics and selection methods developed specifically for neural networks are discussed in Section 5. In Section 6, the chapter is summarized.

## *2.2 Linear Regression*

There are several good references which survey aspects of the variable (feature) selection problem for linear regression [26, 43, 47, 49]. In the linear regression literature features are generally referred to as predictor variables or just variables. In this section, univariate response linear regression is reviewed including standard variable evaluation criteria and selection procedures. An extension to the multivariate response case can be found in Chapter 8 of Anderson [3] and Chapter 15 of Krzanowski [35].

*2.2.1 Univariate Linear Regression Overview.* Univariate response linear regression is a technique which describes the statistical relationship between a response variable and a set of fixed predictor variables. A statistical relationship, in contrast to a functional relationship, is not exact. Statistical relationships are characterized by the tendency for the response variable to change systematically with the set of fixed predictor (regressor) variables, and the tendency for points to scatter around the curve of statistical relationship. Neter, Wasserman, and Kutner [47:27] describe how these characteristics are embodied in a regression model by postulating:

- There is a probability distribution of the response variable for each level of the predictor variables.
- The means of these probability distributions of the response variable vary in some systematic fashion with the predictor variables.

A discussion of linear regression assuming randomly distributed predictor variables with a multivariate normal distribution can be found in both Thompson [73] and Anderson [3].

Consider the linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \qquad \boldsymbol{\varepsilon} \sim \mathbf{N}_n(\mathbf{0}, \sigma^2\mathbf{I}_n) \tag{3}$$

where, $\mathbf{y}$ is an $n$-dimensional vector of responses, and $\mathbf{X}$ is an $n$ by $k$ matrix with $k-1$ columns of fixed independent predictor variables and a column of 1's for the constant or bias term. Also, $\boldsymbol{\beta}$ is a $k$-dimensional vector of unknown variable coefficients. The predicted or fitted vector of $\mathbf{y}$, denoted $\hat{\mathbf{y}}$, is defined

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{b}, \tag{4}$$

where $\mathbf{b}$ is a $k$-dimensional vector of estimated parameters for $\boldsymbol{\beta}$.

The method of least squares is used more extensively than any other estimation procedure for determining regression model coefficients [46:12]. This method requires minimization of the sum of squared errors, SSE, given as

$$\text{SSE} = (\mathbf{y} - \hat{\mathbf{y}})'(\mathbf{y} - \hat{\mathbf{y}})$$

now substituting for $\hat{\mathbf{y}}$ using Equation 4 gives

$$\text{SSE} = (\mathbf{y} - \mathbf{Xb})'(\mathbf{y} - \mathbf{Xb})$$

To find the least squares estimator, partial derivatives of SSE are taken with respect to b and set equal to 0, defining a set of normal equations. In matrix terms, these normal equations are:

$$(\mathbf{X}'\mathbf{X})\mathbf{b} = (\mathbf{X}'\mathbf{y}), \tag{5}$$

where $\mathbf{X}'\mathbf{X}$ is a $k$-dimensional matrix and $\mathbf{X}'\mathbf{y}$ is a $k$-dimensional vector. Now, assuming $(\mathbf{X}'\mathbf{X})^{-1}$ exists, Equation 5 is solved for b using matrix algebra giving the least squares estimate:

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{y}) \tag{6}$$

Regardless of the distribution of the errors $\boldsymbol{\varepsilon}$, the method of least squares provides unbiased point estimators with minimum variance among all unbiased linear estimators [47:52]. However, for confidence intervals and most statistical hypothesis tests, it is necessary to assume that the errors are independently distributed as given in Equation 3.

In neural network terms, the vector $\mathbf{y}$ is the $n \times 1$ vector of observations of desired outputs whereas the vector $\hat{\mathbf{y}}$ is the $n \times 1$ vector of actual outputs or trained neural network outputs. The matrix $\mathbf{X}$ is the data matrix of measured feature vectors, including the bias term which is equal to one for each feature vector. The vector $\boldsymbol{\beta}$ is somewhat analogous to the vector of unknown optimal neural network weight parameters, and the vector of estimated coefficients b is somewhat analogous to the vector of trained or estimated neural network weight parameters. Also, SSE is the squared error function which is minimized in the standard backpropagation algorithm.

*2.2.2 Selection Criteria.* In this section, six of the common variable selection criteria used with linear regression are reviewed. The first two, $R_p^2$ and $R_a^2$, are measures of the proportionate reduction in the total variation of $\mathbf{y}$ associated with a subset of $p$ predictor variables. These metrics can be used to evaluate the quality of a regression model's fit to the present data. The third and fourth criterion, $C_p$ and Press$_p$, are measures of a regression model's predictive error for a subset of $p$ predictor variables. These criteria are computed using a validation data set which is independent of the data set used to estimate the regression parameters. $C_p$ and Press$_p$ are used to assess the quality of future prediction for a subset of candidate variables. The last two selection criteria are the Akaike and Swartz information criteria. These criteria are based on maximizing the theoretic information content of a variable subset.

The coefficient of multiple determination, $R_p^2$, measures the proportionate reduction of total variation in $\mathbf{y}$ associated with a particular set of $p$ predictor variables. The total sums of squares SSTO which is constant regardless of which predictors are used, the regression sums of squares for $p$ predictor variables SSR$_p$, and the sum of squared errors for $p$ predictor variables SSE$_p$ are all needed to define $R_p$.

$$\begin{aligned} \text{SSTO} &= (\mathbf{y}-\bar{\mathbf{y}})'(\mathbf{y}-\bar{\mathbf{y}}) \\ \text{SSR}_p &= (\hat{\mathbf{y}}-\bar{\mathbf{y}})'(\hat{\mathbf{y}}-\bar{\mathbf{y}}) \qquad (7) \\ \text{SSE}_p &= (\mathbf{y}-\hat{\mathbf{y}})'(\mathbf{y}-\hat{\mathbf{y}}) \qquad (8) \end{aligned}$$

Now, the coefficient of multiple determination can be defined as

$$R_p^2 = \frac{\text{SSR}_p}{\text{SSTO}} = 1 - \frac{\text{SSE}_p}{\text{SSTO}}$$

where the subscript $p$ indicates that only $p$ predictor variables have been used from a 'superset' of $q$ candidate predictors, where $p \leq q$.

For classical least squares linear regression, presented in Section 2.2.1, SSTO = SSR$_p$ + SSE$_p$. The criterion $R_p^2$ does not correct for the number of variables in the model; therefore, it is maximized when all $q$ predictor variables are used. The criterion is useful for comparing several subsets of equal size. The subset with the largest value of $R_p^2$ is the subset which is associated with reducing the largest proportion of the total variation in $\mathbf{y}$. When subsets are not of equal size, $R_p^2$ can be subjectively inspected to identify variables which do not substantially increase in $R_p^2$.

The $R_a^2$ criterion is similar to the $R_p^2$ criterion, except that it is adjusted for the number of variables in the regression model. It is defined as

$$R_a^2 = 1 - \frac{\frac{\text{SSE}_p}{n-p}}{\frac{\text{SSTO}}{n-1}}$$

Using the fact that mean squared error for $p$ predictor variables $\text{MSE}_p$ is defined as

$$\text{MSE}_p = \frac{\text{SSE}_p}{n-p}, \tag{9}$$

Now, $R_a^2$ can be written as

$$R_a^2 = 1 - \frac{MSE_p}{(\frac{\text{SSTO}}{n-1})}$$

Maximizing $R_a^2$ is equivalent to minimizing the mean squared error for $p$ variables $MSE_p$, therefore, a good feature subset will be associated with a large value of $R_a^2$. The relationship between $R_a^2$ and $R_p^2$ is

$$R_a^2 = R_p^2 - \frac{p-1}{n-p}\frac{\text{SSE}_p}{\text{SSTO}}$$

Mallows suggests a criterion based on minimizing the mean squared error of prediction MSEP [42]. The definition of MSEP for $p$ predictor variables is the same as $\text{MSE}_p$ in Equation 9, except that now the variables $\mathbf{y}$, $\hat{\mathbf{y}}$, and $n$ correspond to an independent validation data set (different from the data set used to estimate the regression parameters).

A standardized MSEP criterion, $\Gamma_p$, is defined as the ratio of the reduced model's MSEP over the full model's true error variance, $\sigma^2$. This criterion is meant to find a reduced model with $p$

predictor variables that provides a similar MSEP to the full model. Here, the full model is assumed to be unbiased, so the full model's MSEP is an unbiased estimator of $\sigma^2$.

Mallow's $C_p$ criterion is an estimator of $\Gamma_p$ and is defined as

$$C_p = \frac{\text{SSEP}_p}{\text{MSEP}} - (n - 2p)$$

where $\text{SSEP}_p$ is the sum of squared errors for $p$ predictor variables on the validation data set, and MSEP is the mean square prediction error for the full model on the validation data set. If there is no bias in the reduced model of $p$ predictor variables, then $\text{SSEP}_p \approx \text{SSEP}$ for the full model, and the expected value of $C_p$ is approximately $p$. For a good feature subset, the $C_p$ criterion should be small and as close to $p$ as possible, indicating a small prediction bias associated with the reduced regression model of $p$ predictors.

The $\text{PRESS}_p$ selection criterion proposed by Allen is based on minimizing the sum of squared deleted residuals [2]. Let $\mathbf{d}$ be the $n$-dimensional vector of deleted residuals, where the deleted residual $d_i$ is the prediction error for observation $i$ when a regression model is fit without the $i$th observation. The $\text{PRESS}_p$ statistic for a model with $p$ predictor variables is formed by the sum of the squared deleted residuals for that model.

$$\text{PRESS}_p = \mathbf{d}\mathbf{d}'$$

An equivalent expression for $d_i$ can be used which makes it unnecessary to re-estimate the regression model for each $d_i$ [47:451]. The equivalent expression is

$$d_i = \frac{e_i}{1 - h_{ii}}$$

where $e_i$ is the ordinary residual with no observations deleted, and $h_{ii}$ is the $i$th element along the diagonal of the matrix $\hat{\mathbf{H}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$. Models with the lowest $PRESS_p$ for a subset of sise $p$ fit well in a predictive sense. This criterion is subjective when used to discriminate between variable subsets of different sises $p$.

Akaike proposed a criterion developed for statistical model identification based on information theoretic considerations [1]. The criterion can be defined as

$$AIC_p = \log[f(\mathbf{y};\mathbf{X},\mathbf{b})] - p$$

where $p$ is the number of unknown parameters, and $f(y;\mathbf{X},\mathbf{b})$ is the probability density function of $y$ evaluated at $\mathbf{b}$ (maximum likelihood estimates of the $p$ unknown parameters of the subset model). A good feature subset is identified by a maximum $AIC$.

Schwartz proposes a Bayesian version of $AIC$ which is also maximized when used for feature selection [71]. It is defined as

$$\log[f(y;\mathbf{X},\mathbf{b})] - \frac{1}{2}p\log n$$

where $f(y;\mathbf{X},\mathbf{b})$ is the probability density function of $y$ evaluated at $\mathbf{b}$ (maximum likelihood estimates of the $p$ unknown parameters of the subset model), and $n$ is the number of samples. This criterion is better suited than Akaike's to selecting lower-dimensional models. When $n$ is large, the two procedures may produce results which differ greatly [71:463].

The selection criteria described in this section are methods for measuring the relative worth of one variable subset compared to another. By themselves, however, they do not indicate which variables should be retained in a linear regression model. These selection criteria in concert with the selection methodologies described next can be used to determine what variables to retain.

*2.2.3 Selection Methodologies.* Generally, a variable selection procedure involves both a criterion (metric) to evaluate or rank variable subsets, and a methodology to select the *best* subsets. Variable selection involves selecting the $p$ best variables from among $q$ candidate variables, where $p \leq q$. Some of the well known methodologies for selecting the best variable subset are discussed in this section. The well known methodologies include: explicit enumeration of all subset regressions, best $k$ subset regressions, and sequential search of subsets. Two other methodologies are reviewed: ridge regression and principal components regression.

The regression of all subsets requires an explicit enumeration of all $2^q - 1$ possible subsets. When this approach is used, one of the selection criterion from Section 2.2.2 is used to subjectively discriminate between variable subsets. Sometimes, a small subset of the best regression models is selected for further detailed examination. This method is the most computationally intensive of the selection methods described. Time-saving algorithms, which evaluate substantially fewer subsets, are described next.

Best $k$ subsets algorithms have been developed which give the best $k$ subsets according to a given criterion. Furnival and Wilson propose a branch and bound algorithm which uses product inverse matrices and a sophisticated sequence of pivots or Gaussian eliminations [18]. The algorithm evaluates the $\mathrm{SSR}_p$ at each step against some bound to determine the next pivot. Furnival and Wilson's algorithm provides the $k$ best regressions for each subset size $p$.

Sequential algorithms are substantially less computational; however, there is no guarantee that these methods provide the best variable subset. Three algorithms for sequential selection are: forward selection, backward selection, and stepwise selection. These algorithms discriminate between feature subsets using a measure of the predictive or explanatory capability of the regression function. A measure of a regression function's predictive capabilities for a specific subset is given by the metrics described in the previous section. A conditional measure of the additional explanatory contribution of a variable or a set of variables is embodied in the $F$-statistic given in a partial $F$-test.

The partial $F$-test is used to test the significance of one or more variables. The linear regression model in Equation 3 is used with the partial $F$-test. It is given again for reference as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathbf{N}_n(\mathbf{0}, \sigma^2\mathbf{I}_n),$$

where $\mathbf{y}$ is an $n$-dimensional vector of responses; $\mathbf{X}$ is an $n \times k$ matrix of regressor variables consisting of $k - 1$ independent variables and a column of 1's for the constant term; and $\boldsymbol{\beta}$ is a $(k + 1)$-dimensional vector of unknown variable coefficients. The partial $F$-test is associated with the null hypothesis:

$$\mathbf{H_0}: \quad \boldsymbol{\beta}_{p+1} = \cdots = \boldsymbol{\beta}_k = 0,$$

where $p$ is the number of parameters hypothesized to be in the model, and $k - p$ are the number of parameters hypothesized to be 0. When testing the significance of just a single variable, then $k - p = 1$ or $p = k - 1$ in $\mathbf{H_0}$. The associated test statistic, denoted $F^*$, is defined:

$$F^* = \frac{\frac{\mathrm{SSE}_p - \mathrm{SSE}_k}{(n-p)-(n-k)}}{\frac{\mathrm{SSE}_k}{n-k}} \tag{10}$$

Under $\mathbf{H_0}$:

$$F^* \sim F_{k-p,n-k}$$

When $F^* \leq F_{k-p,n-k}$, the null hypothesis can not be rejected which means the reduced model with some parameters hypothesized to be zero is accepted. The variables which are not in the reduced model correspond to the parameters which are hypothesized to be zero. Otherwise, if $F^* > F_{k-p,n-k}$, then the null hypothesis can be rejected. This means that the full model is accepted and no variables are removed.

The sequential selection algorithms presented in this section are based on the $F$-statistics computed for the partial $F$-test. The strategy is to apply successive partial $F$-tests to test the explanatory contribution of a variable to the total sum of squares.

The forward selection procedure starts with no variables in the model and increases the number of variables one at a time. At each step, partial $F$-statistics are computed. If the null hypothesis can not be rejected, then the algorithm terminates at that number of variables in the model.

### Forward Selection

1. $p = 0$

2. Compute all regression models for size $k = p + 1$ which include all previously selected $p$ variables in the model

3. Select as a candidate for entry, the new variable associated with the model having the highest $F^*$ statistic computed using Equation 10

4. Perform partial $F$-test described beginning on page 23

5. If $F^*_{k-p,n-k} > F_{k-p,n-k}$, allow candidate variable to enter model, set $p = p + 1$ and go to Step 2
   Otherwise, go to Step 6

6. Stop

The backward selection procedure starts with all the variables in the model and decreases the number of variables one at a time. At each step, partial $F$-statistics are computed. If the null hypothesis is rejected at any step, the algorithm terminates. Advantages of a backward selection procedure over a forward selection procedure are discussed in Mantel [43]. These advantages include economy of effort and, potentially, better variable selection when correlated variables are present. Because of these advantages, the backwards sequential selection algorithm is used in the selection procedures presented in Chapters V and VI.

### Backward Selection

1. $k = q$, where $q$ is the total number of candidate variables

2. $p = k - 1$

3. Compute all regression models for size $p$ which do not include any previously eliminated variables

4. Select as a candidate variable for elimination, the variable which when removed produces the model with the lowest $F^*$ statistic computed using Equation 10

5. Perform partial $F$-test described beginning on page 23

6. If $F^*_{k-p,n-k} > F_{k-p,n-k}$, go to Step 7 and do not eliminate candidate variable
   Otherwise, eliminate candidate variable, set $k = k - 1$, and go to Step 2

7. Stop

Stepwise selection procedures include forward and backward stepwise procedures. They are, essentially, modifications of the forward and backward selection procedures. The forward stepwise procedure tests to see if any other variables should be eliminated after each iteration of the algorithm. The backward stepwise procedure tests to see if any other variables should be included after each iteration of the algorithm. The forward stepwise algorithm is illustrated here. Like the forward selection algorithm, the forward stepwise selection algorithm starts with no variables in the model.

### Forward Stepwise Selection

1. $p = 0$

2. Compute all regression models for size $k = p + 1$ which include all previously selected $p$ variables in the model

3. Select as a candidate for entry, the new variable associated with the model having the highest $F^*$ statistic computed using Equation 10

4. Perform partial $F$-test described beginning on page 23

5. If $F^*_{k-p,n-k} > F_{k-p,n-k}$, allow candidate variable to enter model, and go to Step 6
   Otherwise, go to Step 9

6. Compute all regression models of size $p$ which include only variables that are currently in the model

7. Select as a candidate for elimination, the variable which when removed produces the model with the lowest $F^*$ statistic

8. If $F^*_{k-p,n-k} > F_{k-p,n-k}$, set $p = p + 1$, and do not eliminate candidate variable
   Otherwise, eliminate the candidate variable and go to Step 2

9. Stop

Pope and Webster, and Krishnaiah point out problems with the use of the $F$-statistic for the forward, backward, and stepwise sequential procedures [36, 49]. Krishnaiah recommends not using these procedures for the selection of variables, since, in general, the necessary conditions are not met for $F^*$ to be distributed as an $F$ distribution [36:814] [49:331-332]. Instead, Krishnaiah recommends using the overall $F$ test and methods based on all possible regressions or finite intersection tests (FIT). The FIT, developed by Krishnaiah, involves using a multivariate $F$-distribution for simultaneously testing whether a *set of variable coefficients* are zero [65]. Details for applying the FIT for either univariate or multivariate regression can be found in Schmidhammer [65].

Another technique which can be utilized for feature selection is ridge regression. Ridge regression is a biased regression technique primarily designed as a remedy for multicollinearity. The technique introduces a biasing constant, $c$, where $c \geq 0$, into the least squares normal equation, Equation 5, to give

$$(\mathbf{X}'\mathbf{X} + c\mathbf{I})\mathbf{b}^R = \mathbf{X}'\mathbf{y}$$

A simple way of choosing $c$ is to increase $c$ gradually and plot a ridge trace. A ridge trace is a plot of the regression coefficients against $c$. Variables which are candidates for elimination are associated with unstable ridge traces (details can be found in Neter Wasserman and Kutner [47:414-417]) and coefficients close to zero.

Principal components regression is yet another technique which can be utilized for feature selection. In principal components regression, the original variables are transformed using linear combinations suggested by the eigenvectors of the covariance matrix. Transformed variables associated with very small eigenvalues can be identified as candidates for elimination, since they are not associated with a very big portion of the original variables' variance. There are two problems associated with this method: difficulty in interpreting the transformed variable. and lack of a quantitative stopping rule to indicate the number of transformed variables to drop.

### *2.3 Nonlinear Regression*

There is a limit to what can be adequately approximated by a linear model, even after exploiting transformations of the dependent or independent variables. When this is the case, a nonlinear regression model is usually considered. This section reviews univariate nonlinear regression and standard statistical selection criteria. Multivariate response extensions to the univariate case are presented in Chapter 5 of Gallant [20].

*2.3.1 Univariate Nonlinear Regression Overview.* Univariate response nonlinear regression is analogous to linear regression: it is a technique which describes the statistical relationship between a response variable and a $q$-dimensional set of fixed predictor variables. The predictor variables are treated as fixed known constants and not as random variables [20:2]. However, it is possible to consider the "random predictor variables case" of nonlinear regression as a special case of the "fixed predictor variables theory [20:247]." The discussion of the nonlinear regression model in this section is taken from both Gallant's [20] and Seber and Wild's [66] presentations.

Consider the univariate nonlinear regression model

$$\mathbf{y} = f(\mathbf{X}, \boldsymbol{\theta}) + \boldsymbol{\varepsilon} \qquad \boldsymbol{\varepsilon} \sim \mathbf{N}_n(\mathbf{0}, \sigma^2\mathbf{I}_n)$$

where, $\mathbf{y}$ is an $n$-dimensional vector of responses, $\mathbf{X}$ is an $n \times s$ matrix with $q$ columns of predictor variables and a column of 1's for the bias term, $\boldsymbol{\theta}$ is an $s$-dimensional vector of unknown parameters, and $f(\mathbf{X},\boldsymbol{\theta})$ is the response function. The functional form hypothesized for $f(\mathbf{X},\boldsymbol{\theta})$ is normally chosen based on the problem at hand or experience.

The predicted value of $\mathbf{y}$, denoted $\hat{\mathbf{y}}$, is defined

$$\hat{\mathbf{y}} = f(\mathbf{X},\hat{\boldsymbol{\theta}})$$

where $\hat{\boldsymbol{\theta}}$ is the least squares estimate for $\boldsymbol{\theta}$. The least squares estimator $\hat{\boldsymbol{\theta}}$ is found by minimizing the sum of squared errors SSE

$$\mathrm{SSE} = [\mathbf{y} - f(\mathbf{X},\hat{\boldsymbol{\theta}})]'[\mathbf{y} - f(\mathbf{X},\hat{\boldsymbol{\theta}})]$$

with respect to $\hat{\boldsymbol{\theta}}$.

In neural network terms, the vector $\mathbf{y}$ is the $n \times 1$ vector of observations of desired outputs whereas the vector $\hat{\mathbf{y}}$ is the $n \times 1$ vector of actual outputs or trained neural network outputs. The matrix $\mathbf{X}$ is the data matrix of measured feature vectors, including the bias term which is equal to one for each vector. The vector $\boldsymbol{\theta}^*$ is analogous to the vector of unknown optimal neural network weight parameters, and the vector $\hat{\boldsymbol{\theta}}$ is the vector of trained or estimated neural network weight parameters. Also, SSE is the squared error function which is minimized in standard backpropagation. In Chapter V, the neural network model will be posed in the framework of a nonlinear regression statistical model.

To find the parameters which minimize SSE, the nonlinear function $f(\mathbf{X},\boldsymbol{\theta})$ is expanded in a Taylor series about $\boldsymbol{\theta} = \boldsymbol{\theta}^*$, retaining only the linear terms as [46:427]:

$$f(\mathbf{X},\boldsymbol{\theta}) \approx f(\mathbf{X},\boldsymbol{\theta}^*) + \mathbf{F}(\mathbf{X},\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*),$$

where

$$\mathbf{F}(\mathbf{X},\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}'} f(\mathbf{X},\boldsymbol{\theta}) \tag{11}$$

and $\mathbf{F}(\mathbf{X}, \boldsymbol{\theta}^*)$ represents $\mathbf{F}$ evaluated at $\boldsymbol{\theta} = \boldsymbol{\theta}^*$. The truncated Taylor series representation is essentially a linearization of $f(\mathbf{X}, \boldsymbol{\theta})$ in the neighborhood of the unknown parameter $\boldsymbol{\theta}^*$. With this form of $f(\mathbf{X}, \boldsymbol{\theta})$, several methods, such as the Gauss-Newton method, exist for computing the least squares parameters, $\hat{\boldsymbol{\theta}}$ [20:26-46].

In general, estimators for $\boldsymbol{\theta}$ are not unbiased [46:426]. The properties of unbiasedness and minimum variance are only approached asymptotically or in the limit when $n$ the number of observations is large [46:426]. The distributional properties of normal-error least squares parameters have practical importance, since the assumed distributional properties are used in hypothesis testing to make inferences about the parameters. With the assumption of normal errors and certain regularity conditions, $\hat{\boldsymbol{\theta}}$ has approximately an $s$-dimensional multivariate normal distribution defined as

$$\hat{\boldsymbol{\theta}} \sim \mathbf{N}_s[\boldsymbol{\theta}, \mathbf{s}^2(\mathbf{F}'\mathbf{F})^{-1}]$$

where $\mathbf{F}$ is defined in Equation 11, and

$$\mathbf{s}^2 = \frac{\text{SSE}}{n - s}$$

is an estimate of the error variance $\sigma^2$ corresponding to $\hat{\boldsymbol{\theta}}$.

In practice, the matrix $\hat{\mathbf{C}}$ is used to approximate $(\mathbf{F}'\mathbf{F})^{-1}$, assuming $(\mathbf{F}'\mathbf{F})^{-1}$ exists, where

$$\hat{\mathbf{C}} = (\mathbf{F}(\hat{\boldsymbol{\theta}})'\mathbf{F}(\hat{\boldsymbol{\theta}}))^{-1} \tag{12}$$

The assumption of normal errors and certain regularity conditions also gives

$$\frac{(n - s)\mathbf{s}^2}{\sigma^2} \sim \chi^2_{n-s} \tag{13}$$

Gallant rigorously presents the regularity conditions used to make these distributional assumptions [20]. The conditions are summarized below [20:19-21]:

1. The sequence of observations on predictor variables $\mathbf{x}$ must behave properly as $n$ tends to infinity. Proper behavior is obtained when:

   - The predictor variable observations are chosen by random sampling.
   - The predictor variable observations are chosen by disproportionate replication of a fixed set of points.

2. The response function $f(\mathbf{X}, \boldsymbol{\theta})$, as well as it's first and second partial derivatives, must be continuous in $(\mathbf{X}, \boldsymbol{\theta})$.

3. The identification condition holds which requires that the $\lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} [f(\mathbf{x}_i, \boldsymbol{\theta}) - f(\mathbf{x}_i, \boldsymbol{\theta}^*)]^2$ has a unique minimum at $\boldsymbol{\theta} = \boldsymbol{\theta}^*$, where $\mathbf{x}_i$ is the $i$th predictor variable.

4. The rank qualification condition holds which requires that the $\lim_{n\to\infty} n^{-1} \sum_{i=1}^{n} \mathbf{F}'(\mathbf{x}_i, \boldsymbol{\theta}^*)\mathbf{F}(\mathbf{x}_i, \boldsymbol{\theta}^*)$ be nonsingular.

*2.3.2 Statistical Model Selection Criteria.* In this section, three statistical selection criteria for nonlinear regression are reviewed: the Wald test statistic, the likelihood ratio test statistic, and the Lagrange multiplier test statistic. Seber and Wild discuss the asymptotic equivalence of these statistics [66:571-581]. All three criteria are test statistics which are formed during hypothesis testing. Several detailed examples of applying these test statistics for hypothesis testing are given in Gallant [20:48-100]. In this section, these criteria are presented and reviewed in nonlinear regression terminology for ease of access into the nonlinear regression literature. In Chapter V, these criteria will be redefined for use in the context of a neural network model.

The first model selection criterion is the Wald test statistic $\mathbf{W}$ [20:48]. Define $h(\boldsymbol{\theta})$ as a once differentiable function mapping from $\mathcal{R}^s$ to $\mathcal{R}^k$ with a $s \times k$ Jacobian $\mathbf{H}$ defined as

$$\mathbf{H}(\boldsymbol{\theta}) = \frac{d}{d\boldsymbol{\theta}'} h(\boldsymbol{\theta}), \tag{14}$$

where $s$ is the number of parameters in the full model, and $k$ are the number of parameters hypothesized to be zero in the reduced model. Consider testing the hypothesis $\mathbf{H}_0$ which is given as

$$\mathbf{H}_0 : h(\boldsymbol{\theta}^*) = 0,$$

The Wald test statistic is defined as

$$\mathbf{W} = \frac{h'(\hat{\boldsymbol{\theta}})(\hat{\mathbf{H}}\hat{\mathbf{C}}\hat{\mathbf{H}}')^{-1}h(\hat{\boldsymbol{\theta}})}{k\,\mathbf{s}^2}$$

where $\hat{\mathbf{C}}$ and $\hat{\mathbf{H}} = \mathbf{H}(\hat{\boldsymbol{\theta}})$ were defined previously in Equations 12 and 14, $\mathbf{s}^2$ is for the full model with $s$ parameters, and $k$ is the number of parameters in the hypothesized model. The Wald test statistic is formed using the ratio of two independent chi-squared statistics each divided by its respective degrees of freedom. Under $\mathbf{H}_0$, the numerator, to within approximation error, is a chi-squared statistic with $k$ degrees of freedom which is associated with the null hypothesis (see Gallant for details [20:47-48]). The denominator, to within approximation error, is a chi-square statistic with $n - s$ degrees of freedom, and is associated with the full model [20:47-48]. The chi-squared statistic in the denominator was previously shown in Equation 13.

Under $\mathbf{H}_0$, the Wald statistic is distributed as an $F$-distribution with $k$ numerator and $n - s$ denominator degrees of freedom. When $\mathbf{W}$ exceeds the $\alpha \times 100\%$ critical point of the $F$-distribution with $k$ numerator and $n - s$ denominator degrees of freedom, the hypothesis $H_0$ is rejected. This means that the reduced model with fewer parameters is statistically not equivalent to the full model with all $s$ parameters. Therefore, no parameters would be removed.

The second model selection criterion is the likelihood test statistic, which is analogous to the partial $F$-test discussed in the previous section. The likelihood test statistic $\mathbf{L}$ is also used to test $\mathbf{H}_0 : h(\boldsymbol{\theta}^*) = 0$, where $h(\boldsymbol{\theta}^*)$ is defined to be a subset of $k$ parameters from the full set of $s$ parameters. It is defined as

$$\mathbf{L} = \frac{(\mathrm{SSE}_{s-k} - \mathrm{SSE}_s)\,/(\,(n-(s-k)) - (n-s)\,)}{(\mathrm{SSE}_s)\,/(n-s)}, \tag{15}$$

where $\mathrm{SSE}_{s-k}$ corresponds to the sum of squared errors associated with fitting the model under the null hypothesis, and $\mathrm{SSE}_s$ corresponds to the sum of squared errors associated with the full model. Under $\mathbf{H}_0$, $\mathbf{L}$ is distributed as an $F$ distribution with $k$ and $n-s$ degrees of freedom for the numerator and denominator, respectively. When the $\mathbf{L}$ exceeds the $\alpha \times 100\%$ critical point of the $F$-distribution with $k$ numerator and $n-s$ denominator degrees of freedom, the hypothesis $H_0$ is rejected. This means that the reduced model with fewer parameters is statistically not equivalent to the full model with all $s$ parameters. Therefore, no parameters would be removed.

The third model selection criterion is the Lagrange multiplier or efficient score test statistic. There are two versions of this test statistic discussed in Gallant [20:85-97]. The Lagrange multiplier test statistic is also used to test $\mathbf{H}_0 : h(\boldsymbol{\theta}^*) = 0$, where $h(\boldsymbol{\theta}^*)$ is defined to be a subset of $k$ parameters from the full set of $s$ parameters. Let SSE be minimized for the full model subject to the null hypothesis $h(\boldsymbol{\theta}^*) = 0$. This is a constrained minimization. The resulting estimator of the constrained minimization is denoted $\tilde{\boldsymbol{\theta}}$.

Now $\tilde{\boldsymbol{\theta}}$ is used as a starting value and one 'unconstrained' Gauss-Newton step is taken away from $\tilde{\boldsymbol{\theta}}$, presumably towards $\hat{\boldsymbol{\theta}}$ which is the least squares estimator for the unconstrained minimization of SSE. The Gauss Newton step, denoted $\tilde{\mathbf{D}}$ is defined in Gallant [20:85] as

$$\tilde{\mathbf{D}} = \left(\tilde{\mathbf{F}}'\tilde{\mathbf{F}}\right)^{-1} \tilde{\mathbf{F}}' \left[\mathbf{y} - f(\tilde{\boldsymbol{\theta}})\right],$$

where $\tilde{\mathbf{F}} = \mathbf{F}(\tilde{\boldsymbol{\theta}})$ as previously defined in Equation 11. If $\mathbf{H}_0$ is true, then $\tilde{\mathbf{D}}$ will be small. Conversely, if $\mathbf{H}_0$ is false, then $\tilde{\mathbf{D}}$ will be large. The two forms of Lagrange test statistics, $R_1$ and $R_2$, are based on a measure of $\tilde{\mathbf{D}}$. They are defined in Gallant [20:86] as:

$$\begin{aligned} R_1 &= \frac{\tilde{\mathbf{D}}'\left(\tilde{\mathbf{F}}'\tilde{\mathbf{F}}\right)\tilde{\mathbf{D}}/k}{\mathrm{SSE}(\hat{\boldsymbol{\theta}})/(n-s)} \\ R_2 &= \frac{n\tilde{\mathbf{D}}'\left(\tilde{\mathbf{F}}'\tilde{\mathbf{F}}\right)\tilde{\mathbf{D}}}{\mathrm{SSE}(\tilde{\boldsymbol{\theta}})}, \end{aligned}$$

where SSE($\hat{\theta}$) represents the minimum sum of squared errors when fitting the full model with no constraints, and SSE($\tilde{\theta}$) represents the constrained minimum sum of squared errors when fitting the full model subject to $\mathbf{H}_0$. With $R_1$, the Lagrange multiplier test rejects $\mathbf{H}_0$ when $R_1$ exceeds $F_\alpha = F^{-1}_{1-\alpha;k,n-s}$. With $R_2$, the Lagrange multiplier test rejects $\mathbf{H}_0$ when $R_2$ exceeds $d_\alpha$ which is defined

$$d_\alpha = \frac{nF_\alpha}{(n-s)/k + F_\alpha}$$

*2.3.3 Summary of Statistical Model Selection Criteria.* In Summary, there are three model selection hypothesis tests used in feature selection for nonlinear regression: the Wald hypothesis test, the likelihood ratio test and the Lagrange multiplier tests. The Wald test statistic has both advantages and disadvantages. The advantage is that only the unconstrained/full model must be estimated for this criterion. However, the disadvantage is that Wald test statistic can be seriously affected by parameter effects curvatures of the error surface which are not accounted for in the linearized approximation used for $f(\mathbf{X},\theta)$ [66:200-220]. The likelihood ratio test statistic and the Lagrange multiplier test statistics are not affected by the parameter effects curvatures [66:200]. Between the two versions of the Lagrange multiplier test, the first test $R_1$ is always more powerful than $R_2$, although this increase in power is sometimes negligible [20:88-89]. In practice, $R_2$ is more commonly used than $R_1$, since $R_2$ requires just one minimization of SSE rather than two, making $R_2$ easier to compute [20:86-87]. When comparing the likelihood ratio test statistic and the two versions of the Lagrange multiplier test statistic, the likelihood test statistic is always the more powerful [20:89].

The last two sections reviewed feature selection topics related to linear and nonlinear regression. The next section reviews feature selection topics related to discriminant analysis.

## *2.4 Discriminant Analysis*

In discriminant analysis problems, the practitioner is concerned with classifying a pattern or an object into one of $K$ classes. A pattern or an object is represented by a vector of features which ideally contain discriminatory information between the classes. There are many algorithms available for doing discriminant analysis. The vast number of algorithms makes it impossible to provide a concise overview of these algorithms in this document. Devijer and Kittler, Dillon and Goldstein, and James are all good references on discriminant analysis and pattern recognition algorithms [14, 15, 29].

Feature selection in these applications is similar to linear regression, since, ideally, feature subsets would be evaluated based on a measure of the classification function's accuracy. In linear regression, calculating this measure is very simple, but for discriminant analysis it can be prohibitively time consuming. Therefore, feature selection is often carried out by trying to maximize an alternative measure. *This* alternative *measure is* hopefully related closely to the error rate of the resulting classifier [29:127]. There are several good references which review feature evaluation criteria and feature selection methodologies in the context of pattern recognition and discriminant analysis [8, 14, 15, 29, 74]. In the remainder of this section, standard distance metrics used for evaluating or ranking features and feature selection methodologies are reviewed.

*2.4.1 Selection Metrics.* A common need for all feature selection procedures is to have an evaluation function for measuring the saliency or potency of features. This section reviews non-probabilistic and probabilistic feature evaluation metrics. The non-probabilistic feature evaluation metrics are reviewed first, since they are relatively simple to calculate. After a discussion of the non-probabilistic metrics, the more sophisticated, often computationally burdensome, probabilistic metrics are reviewed.

Non-probabilistic feature evaluation metrics are used as an alternative to measuring the error rate or a sophisticated probabilistic measure of the classifier's performance. Generally, these metrics are a measure of distance between classes. The non-probabilistic feature evaluation metrics are based on the rationale that classes should be maximally separated in the feature space. The larger the average separation between classes, the better the feature subset. While non-probabilistic distance metrics are somewhat unsophisticated compared to the probabilistic metrics, they are relatively easy to calculate.

Generally, these types of metrics attempt to maximize the between class (interclass) distances while minimizing the within class (intraclass) distances. An estimated matrix of the between class distances, $\hat{\mathbf{S}}_b$, is defined as

$$\hat{\mathbf{S}}_b = \sum_{k=1}^{K} \hat{P}(C_k)(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})',$$

where $\mathbf{m}_k$ is a $M$-dimensional mean vector of the $k$th class of $M$-dimensional training vectors, $\mathbf{m}$ is a $M$-dimensional mean vector of all the training vectors, $M$ is the number of features in a training vector, and $\hat{P}(C_k)$ is the estimated prior probability of class $k$, denoted as $C_k$. An estimated matrix of the within class distances, $\hat{\mathbf{S}}_w$, is defined as

$$\hat{\mathbf{S}}_w = \sum_{k=1}^{K} \hat{P}(C_k) N_k{}^{-1} \sum_{i=1}^{N_i} (x_{ki} - \mathbf{m}_k)(x_{ki} - \mathbf{m}_k)',$$

where $x_{ki}$ is the $i$th training vector from the $k$th class and $N_k$ is the number of training vectors in class $k$. Two possible metrics $D(\mathbf{x})$ which are maximized to find a good subset of features are

$$D(\mathbf{x}) = \frac{\mathrm{tr}\mathbf{S}_b}{\mathrm{tr}\mathbf{S}_w}$$

$$D(\mathbf{x}) = \frac{|\,\mathbf{\Sigma}\,|}{|\,\mathbf{S}_w\,|}$$

where $\mathbf{\Sigma}$ is equal to $\mathbf{S}_w + \mathbf{S}_b$. Also, $|\cdot|$ denotes calculation of a determinant. The main criticism of these types of metrics is that they are not closely related to error probability. Two additional drawbacks of these metrics are: discriminatory potential of these metrics depends on the classes having equal covariance matrices; and, pattern classes with equal means may give misleading results.

A general nonlinear metric $D(\mathbf{x})$ which reflects the local probability structure of the data can be maximized to find a good subset of features [14:242]

$$D(\mathbf{x}) = \frac{1}{2}\sum_{k=1}^{K}\hat{P}(C_k)\sum_{l=1}^{K}\hat{P}(C_l)\frac{1}{N_kN_l}\sum_{i=1}^{N_k}\sum_{j=1}^{N_l} d(x_{ki}, x_{lj}),$$

where $d(x_{ki}, x_{lj})$ is the nonlinear distance metric between the $i$th vector in class $k$ and the $j$th vector is class $l$. This nonlinear distance is equal to a constant $H$, if its Euclidean distance is above a threshold $T$, otherwise it is equal to zero. The threshold $T$ represents a safe or effective distance for correctly classifying the two points into separate classes. Methods for determining $T$ are discussed in Devijver and Kittler [8:197-198,242-245]. This nonlinear metric represents a compromise between probabilistic and non-probabilistic feature evaluation metrics.

The following survey of probabilistic distance metrics is primarily based on the discussion in Ben-Basset and Devijver and Kittler [8, 14]. Ben-Basset categorizes probabilistic feature evaluation metrics into three categories by [8:778]:

1. Distance metrics derived from Information measures.
2. Distance metrics derived from distance measures.
3. Distance metrics derived from dependence measures.

The probabilistic feature evaluation metrics are reviewed using this taxonomy.

The first category of probabilistic feature evaluation metrics are the information metrics. Probabilistic information measures are sometimes referred to as uncertainty measures. Given an uncertainty function $u$ and a prior probability vector $\boldsymbol{\pi}$, the information gain, $I(\mathbf{x})$ can be defined in general terms as

$$I(\mathbf{x}) = u(\boldsymbol{\pi}) - \mathrm{E}_{x}[u(\hat{\boldsymbol{\pi}}(\mathbf{x}))]$$

Table 1. Distance Metrics Based on Probabilistic Uncertainty

| Uncertainty Function | Definition |
|---|---|
| Bayes $P_e$ | $u(\hat{\boldsymbol{\pi}}(\mathbf{x})) = [1 - \max\{P(C_1\|\mathbf{x}), \cdots, P(C_k\|\mathbf{x})\}]$ |
| Shannon $H$ | $u(\hat{\boldsymbol{\pi}}(\mathbf{x})) = -\sum_{k=1}^{K} P(C_k\|\mathbf{x}) \log P(C_k\|\mathbf{x})$ |
| Quadratic $Q$ | $u(\hat{\boldsymbol{\pi}}(\mathbf{x})) = \sum_{k=1}^{K} P(C_k\|\mathbf{x})(1 - P(C_k\|\mathbf{x}))$ |
| Daroczy $Q_\alpha$ | $u(\hat{\boldsymbol{\pi}}(\mathbf{x})) = \frac{1}{2^{(1-\alpha)}-1}(\sum_{k=1}^{K} P(C_k\|\mathbf{x})^\alpha - 1)$ |
| $f$-entropy | $u(\hat{\boldsymbol{\pi}}(\mathbf{x})) = \sum_{k=1}^{K} f(P(C_k\|\mathbf{x}))$ |
| Renyi $H_\alpha$ | $u(\hat{\boldsymbol{\pi}}(\mathbf{x})) = \frac{1}{1-\alpha} \log \sum_{k=1}^{K} P(C_k\|\mathbf{x})^\alpha$ |
| This table adapted from [8:780]. | |

where $u(\boldsymbol{\pi})$ is the prior uncertainty and $\mathrm{E}_x[u(\hat{\boldsymbol{\pi}}(\mathbf{x}))]$ is the posterior uncertainty using the vector of posterior probabilities $\hat{\boldsymbol{\pi}}(\mathbf{x})$. The prior uncertainty, $u(\boldsymbol{\pi})$, is independent of $\mathbf{x}$, so an information metric based on $\mathbf{x}$ can be reduced to

$$U(\mathbf{x}) = \mathrm{E}_x[u(\hat{\boldsymbol{\pi}}(\mathbf{x}))]$$

where we want to find the set of features $\mathbf{x}$ which minimizes the uncertainty $U(\mathbf{x})$.

Uncertainty metrics differ by their definition of $u(\hat{\boldsymbol{\pi}}(\mathbf{x}))$. Define the posterior probability of Class $k$ as $P(C_k|\mathbf{x})$ which is used in Table 1 to define several different uncertainty metrics [8:779-780]. All of the uncertainty metrics are bounded above and below by some function of $P_e$, also defined in Table 1 (see Ben-Basset [8:780]). Also, Bayes, Shannon, Quadratic, and Daroczy uncertainty functions all belong to the $f$-entropy family of uncertainty metrics defined in Table 1.

The Bayesian probability of error, $P_e$, is probably the most commonly used feature selection metric. It falls into the category of probabilistic uncertainty metrics. The $P_e$ feature evaluation metric is commonly used whenever the goal is minimizing classifier error rate with features of equal measurement cost. Let the posterior probability of class $k$ for $\mathbf{x}$, $P(C_k|\mathbf{x})$ be defined as

$$P(C_k|\mathbf{x}) = \frac{P(C_k)P(\mathbf{x}|C_k)}{\sum_{k=1}^{K} P(C_k)P(\mathbf{x}|C_k)}$$

where $P(C_k)$ is the prior probability of class $k$ and $P(\mathbf{x}|C_k)$ is the conditional probability function of $\mathbf{x}$ for class $k$. Now, the $P_e$ metric can be given as [8:774]

$$P_e(\mathbf{x}) = \mathrm{E}_{\mathbf{x}}[1 - \max\{P(C_1|\mathbf{x}), \cdots, P(C_k|\mathbf{x})\}] \tag{16}$$

where $\mathbf{x}$ is the vector of features for which $P_e$ is measured.

Ben-Basset discusses several reasons why alternative feature evaluation metrics may be preferred to $P_e$ [8:776-777]. First, the $P_e$ may not be sensitive enough to discriminate between good and better features, because it is based on the most probable class which can be strongly influenced by the prior probabilities $P(C_k)$. Second, sequential selection of features using $P_e$ does not ensure good performance on resulting subsets, even for conditionally independent features [8, 11, 16]. Third, other feature evaluation criterion may perform better when the objective of a myopic sequential selection procedure is to reach a predetermined level of $P_e$ by a minimum number of features. Lastly, computation of $P_e$ may be costly, since it involves integration of the function $\max\{P(C_1|\mathbf{x}), \cdots, P(C_k|\mathbf{x})\}$. Of the four reasons just discussed, Ben-Basset concludes that the most significant reason for avoiding $P_e$ is that it may not be sensitive enough to discriminate between good and better features [8:788]. Even if alternate metrics offer no computational advantage over $P_e$, they should be considered when $P_e$ becomes highly insensitive due to a relatively high prior probability for one class [8].

The second category of probabilistic feature evaluation metrics are the distance metrics. Probabilistic distance metrics are based on distances between probability measures. Measures of probability are prior probability density functions, posterior probability density functions, and conditional density functions. These metrics are also known as discriminatory, separability, or divergence measures. Probabilistic distance metrics can be divided into two classes:

1. Those based on the distance between prior probability density functions and posterior probability density functions of each class.

2. Those based on distances between conditional density functions.

Probabilistic distance functions of the first type are based on the rationale that a feature subset which changes the assessment of the true class probability is a good feature subset. The more drastic the assessed change, the better the feature subset. This type of metric is similar to the probabilistic information metrics except that distance functions are used instead of uncertainty functions. Table 2 presents several different distance measures based on distance between prior and posterior probability density functions for each class [8:783]. For these distance measures, the probability of $\mathbf{x}$, denoted $P(\mathbf{x})$ is defined

$$P(\mathbf{x}) = \sum_{k=1}^{K} P(C_k)P(\mathbf{x}|C_k),$$

where $P(C_k)$ is the prior probability of class $k$, and $P(\mathbf{x}|C_k)$ is the class-conditional probability density function of $\mathbf{x}$ for class $k$.

Probabilistic distance functions of the second type are based on the rationale that the larger the distance between class-conditional density functions $P(\mathbf{x}|C_k)$, the easier it will be to discriminate between classes. For a two class problem, this type of distance function is minimized when $P(\mathbf{x}|C_1) = P(\mathbf{x}|C_2)$, and is maximized when $P(\mathbf{x}|C_1)$ and $P(\mathbf{x}|C_2)$ are orthogonal [8:781]. For a multi-class problem, the distance function can be generalized to be the weighted sum of distances between class-conditioned density functions for all pairs of classes [8:781] [14:261]. The distance function for a multi-class function measures the discriminatory power of the evaluated feature vector over all $K$ classes as

$$D(\mathbf{x}) = \sum_{k=1}^{K}\sum_{l=1}^{K} P(C_k)P(C_l)d_{kl}(\mathbf{x})$$

where $d_{kl}(\mathbf{x})$ is the distance between class $k$ and class $l$'s class-conditioned density functions. A disadvantage of this metric is that one large $d_{kl}$ may dominate $D(\mathbf{x})$ by imposing a ranking which is biased by the two most separable classes [8:782]. Table 3 presents several versions of $d_{kl}$ adapted from Ben-Basset [8:784] and Devijver and Kittler [14:257-258]. Functions of the distance functions in this table serve as lower and upper bounds for $P_e$ [8:784]. Another multi-class distance measure between class-conditional density functions is Matusita's extension of the infinity distance measure, where $D(\mathbf{x})$ is defined as [8:782]

$$D(\mathbf{x}) = \int [\prod_{k=1}^{K} P(\mathbf{x}|C_k)]^{\frac{1}{k}} d\mathbf{x}$$

The third category of probabilistic feature evaluation metrics are the dependence metrics. Probabilistic dependence metrics can also be referred to as correlation metrics. They are natural multi-class feature selection metrics related to both probabilistic information measures and probabilistic distance measures. The probabilistic dependence metrics are based on the rationale that correlation is important between an evaluated feature vector and its true class. The larger the

Table 2. Distance Metrics Based on Prior and Posterior pdfs

| Distance Function | Definition |
|---|---|
| Affinity | $\int[\prod_{k=1}^{K} P(C_k|\mathbf{x})]^{\frac{1}{k}} P(\mathbf{x})d\mathbf{x}$ |
| Bayesian | $\int[\sum_{k=1}^{K} P(C_k|\mathbf{x})^2]P(\mathbf{x})d\mathbf{x}$ |
| Directed Divergence | $\int[\sum_{k=1}^{K} P(C_k|\mathbf{x}) \log \frac{P(C_k|\mathbf{x})}{P(C_k)}]P(\mathbf{x})d\mathbf{x}$ |
| Divergence of Order $\alpha > 0$ | $\frac{1}{\alpha-1} \int[\log \sum_{k=1}^{K} P(C_k|\mathbf{x})^{\alpha} P(C_k)^{(1-\alpha)}]P(\mathbf{x})d\mathbf{x}$ |
| Variance | $\int[\sum_{k=1}^{K} P(C_k)(P(C_k|\mathbf{x}) - P(C_k))^2]P(\mathbf{x})d\mathbf{x}$ |
| This table adapted from [8:783]. | |

Table 3. Distance Metrics Based on Class-Conditioned Density Functions

| Distance Function | Definition $d_{kl}$ |
|---|---|
| Chernoff | $-\log \int (P(\mathbf{x} \mid C_k)^{\alpha} P(\mathbf{x} \mid C_l))^{(1-\alpha)} d\mathbf{x}$ |
| Bhattacharyya (Chernoff where $\alpha = .5$) | $-\log \int (P(\mathbf{x} \mid C_k) P(\mathbf{x} \mid C_l))^{\frac{1}{2}} d\mathbf{x}$ |
| Matusita | $\left\{ \int [\sqrt{P(\mathbf{x} \mid C_k)} - \sqrt{P(\mathbf{x} \mid C_l)}]^2 d\mathbf{x} \right\}^{\frac{1}{2}}$ |
| Kullback-Liebler | $\int [P(\mathbf{x} \mid C_k) - P(\mathbf{x} \mid C_l)] \log \frac{P(\mathbf{x} \mid C_k)}{P(\mathbf{x} \mid C_l)} d\mathbf{x}$ |
| Patrick-Fisher | $\{ \int [P(C_k) P(\mathbf{x} \mid C_k) - P(C_l) P(\mathbf{x} \mid C_l)]^2 d\mathbf{x} \}^{\frac{1}{2}}$ |
| Lissack-Fu | $\int \lvert P(C_k) P(\mathbf{x} \mid C_k) - P(C_l) P(\mathbf{x} \mid C_l) \rvert^{\alpha} P^{(1-\alpha)}(\mathbf{x}) d\mathbf{x}$ |
| Kolomogorov | $\int \lvert P(C_k) P(\mathbf{x} \mid C_k) - P(C_l) P(\mathbf{x} \mid C_l) \rvert \, d\mathbf{x}$ |
| This table adapted from [8:784] and [14:257-258] | |

Table 4. Distance Metrics Based on Probabilistic Dependence

| Distance Function | Definition $R(\mathbf{x})$ |
|---|---|
| Chernoff | $\sum_{k=1}^{K} P(C_k) \{ -\log \int (P(\mathbf{x} \mid C_k)^{\alpha} P(\mathbf{x}))^{(1-\alpha)} d\mathbf{x} \}$ |
| Bhattacharyya (Chernoff where $\alpha = .5$) | $\sum_{k=1}^{K} P(C_k) \left\{ -\log \int (P(\mathbf{x} \mid C_k) P(\mathbf{x}))^{\frac{1}{2}} d\mathbf{x} \right\}$ |
| Matusita | $\sum_{k=1}^{K} P(C_k) \left\{ \int [\sqrt{P(\mathbf{x} \mid C_k)} - \sqrt{P(\mathbf{x})}]^2 d\mathbf{x} \right\}^{\frac{1}{2}}$ |
| Joshi (Kullback-Liebler) | $\sum_{k=1}^{K} P(C_k) \int [P(\mathbf{x} \mid C_k) - P(\mathbf{x})] \log \frac{P(\mathbf{x} \mid C_k)}{P(\mathbf{x})} d\mathbf{x}$ |
| Patrick-Fisher | $\sum_{k=1}^{K} P(C_k) \{ \int [P(\mathbf{x} \mid C_k) - P(\mathbf{x})]^2 d\mathbf{x} \}^{\frac{1}{2}}$ |
| Lissack-Fu | $\sum_{k=1}^{K} P(C_k) \int \lvert P(\mathbf{x} \mid C_k) - P(\mathbf{x}) \rvert^{\alpha} P^{(1-\alpha)}(\mathbf{x}) d\mathbf{x}$ |
| Kolomogorov | $\frac{1}{2} \sum_{k=1}^{K} P(C_k) \int \lvert P(\mathbf{x} \mid C_k) - P(\mathbf{x}) \rvert \, d\mathbf{x}$ |
| This table adapted from [8:785] and [14:261] | |

dependence, denoted $R(\mathbf{x})$, the better the feature vector $\mathbf{x}$. Table 4 presents several versions of $R(\mathbf{x})$ adapted from Ben-Basset [8:785] and Devijver and Kittler [14:261].

*2.4.2 Selection Methodologies.* Selection methodologies used for classification analysis are essentially the same as those used in linear regression. Exhaustive search of all $2^q - 1$ feature subsets for $q$ candidate features is usually impractical for moderately large values of $q$. Even choosing the best subset of size $k$ by exhaustive enumeration is usually impractical, since it involves evaluation of $\frac{q!}{(q-k)!k!}$ subsets. The are a number of optimal and suboptimal search algorithms which are designed to circumvent an exhaustive search procedure. Most search algorithms look for the best features by adding and/or removing features from the current feature set. A forward procedure starts with no features and searches for the features in a "bottom up" manner, whereas a backward procedure begins with all the candidate features and searches in a "top down" manner. A feature evaluation metric is evaluated at each step of these algorithms to determine which features to add or remove at each step. For some of the probabilistic distance metrics displayed in Table 3, computational savings can be realized by exploiting recursive evaluation of the metrics during "bottom up" and "top down" searches [14:265-269]. Genetic or evolutionary algorithms, which will not be discussed, are alternative methods.

Branch and bound is an optimal search algorithm for finding the best subset of size $k$. It is basically a "top down" search which avoids exhaustive search by using the monotonicity property that applies to most feature evaluation metrics [14:207]. The monotonicity property dictates that the error from a set of features is never greater than the error from a subset of those features. This principal can be translated to mean more features implies more information which in turn implies lower error. Monotonicity may allow many feature subsets to be inspected implicitly with no additional computation. Devijver and Kittler provide a detailed presentation of the branch-and-bound algorithm [14:207-214]. When the superset size $k$ is approximately $\frac{q}{2}$, the branch-and-bound algorithm may yield substantial potential savings in computation costs compared to exhaustive enumeration; however, the savings are not as impressive for very small $k$ or $k$ close to $q$ [14:214]. In many cases, the monotonicity assumption used by the branch-and-bound algorithm may be invalid if the statistical structure of the prior class probabilities is not known [74:799]. Van Campenhout discusses the Hughes paradox which documents the phenomena of error being minimized at some finite size feature set [74:796-780].

For many problems, the optimal branch-and-bound algorithm is still computationally impractical. Suboptimal search algorithms which trade off optimality for computational feasibility can be used for these problems. With these algorithms, there is no guarantee that the best feature set will be found. Some suboptimal searches are more sophisticated than others in two respects: the number of computations required, and the number of feature subsets evaluated. There is also no guarantee that more sophisticated algorithms will yield better subsets than less sophisticated algorithms. The next paragraphs summarize several suboptimal search algorithms.

The least sophisticated suboptimal search algorithm is to choose a feature set of size $k$ based on the $k$ best features when measured independently with one of the feature evaluation metrics discussed previously in this chapter. Even if all the features are statistically independent, this method does not guarantee optimality [8, 11, 14, 16].

Sequential forward and backward selection algorithms are similar to those discussed for linear regression in Section 2.2.3. These algorithms allow just one feature to be added or taken away at a time. The main drawback of forward and backward sequential selection algorithms is that they do not allow a feature to be removed or added at a later point in the algorithm. Another drawback is that the total number of features to be selected, $k$, must be known up front, since $k$ serves as a "stopping" mechanism. The algorithms stop when the current number of features in the model, $p$, equals the number to be selected, $k$. The sequential selection algorithms which follow are summarized from Devijer's and Kittler's presentation [14:216-217].

### Forward Sequential Selection

1. $p = 0$
   Set the total number of features to select equal to $k$
   Select any of the feature evaluation metrics described in Section 2.4.1, say $D(\mathbf{x})$

2. Compute $D(\mathbf{x})$ for all feature subsets of size $p + 1$ which include all previously selected $p$ features

3. Select the feature set of size $p + 1$ which maximizes (minimizes) $D(\mathbf{x})$

4. Set $p = p + 1$

5. If $p < k$ go to step 2
   Otherwise, go to step 6

6. Stop, since $k$ features have now been selected

### Backward Sequential Selection

1. $p = q$
   Set the total number of features to select equal to $k$
   Select any of the feature evaluation metrics described in Section 2.4.1, say $D(\mathbf{x})$

2. Compute $D(\mathbf{x})$ for all feature subsets of size $p - 1$ which do not include any previously eliminated features

3. Select the feature set of size $p - 1$ which maximizes (minimizes) $D(\mathbf{x})$

4. Set $p = p + 1$

5. If $p > k$ go to step 2
   Otherwise, go to step 6

6. Stop, since $k$ features have now been selected

Generalized sequential forward or backward selection allow more than one feature to be added or taken away during each iteration [14:217-219]. By taking more than one measurement into consideration, the statistical relationship among potential features is partially taken into consideration. These algorithms are computationally more sophisticated than sequential algorithms, since more feature subsets must be evaluated at each step. These algorithms differ from the non-generalized version of the algorithms primarily in Step 2. Now all feature subsets of size $p + r$ or $p - r$ are evaluated, where $r$ is the number of additional features added or taken away from the feature set. The generalized forward or backward algorithm still does not allow features to be taken away or added later in the algorithm.

Stepwise selection algorithms are sophisticated algorithms which allow one feature to be added and taken away during each iteration of the algorithm. These algorithms partially take into consideration the statistical relationship between feature subsets. Stepwise selection algorithms require more computation than strictly sequential algorithms. The forward sequential algorithm can be adapted to perform forward stepwise selection as follows.

### Forward Stepwise Selection

1. $p = 0$
   Set the total number of features to select equal to $k$
   Select any of the feature evaluation metrics described in Section 2.4.1, say $D(\mathbf{x})$

2. Compute $D(\mathbf{x})$ for all feature subsets of size $p + 1$ which include all previously selected $p$ features

3. Select the feature set of size $p + 1$ which maximizes (minimizes) $D(\mathbf{x})$

4. Compute $D(\mathbf{x})$ for all feature subsets of size $p$ which include previously selected $p + 1$ features

5. If feature subset of size $p$ which maximizes $D(\mathbf{x})$ is not the same subset of size $p$ from step 2, then retain only those $p$ features and go to step 2
   Otherwise, go to step 6

6. Set $p = p + 1$

7. If $p < k$ go to step 2
   Otherwise, go to step 8

8. Stop, since $k$ features have now been selected

Stepwise selection algorithms can also be generalized to allow for more than one feature to be added or taken away (or possibly both added and taken away) at each iteration of the algorithm

[15:220-223]. Further generalization allows the number of features added or taken away to vary at each iteration within the algorithm.

The last three sections have reviewed feature selection topics related to classical linear and nonlinear regression and discriminant analysis. The next section reviews the work which has been done for feature selection specifically in the context of neural networks.

## *2.5 Feedforward Neural Networks*

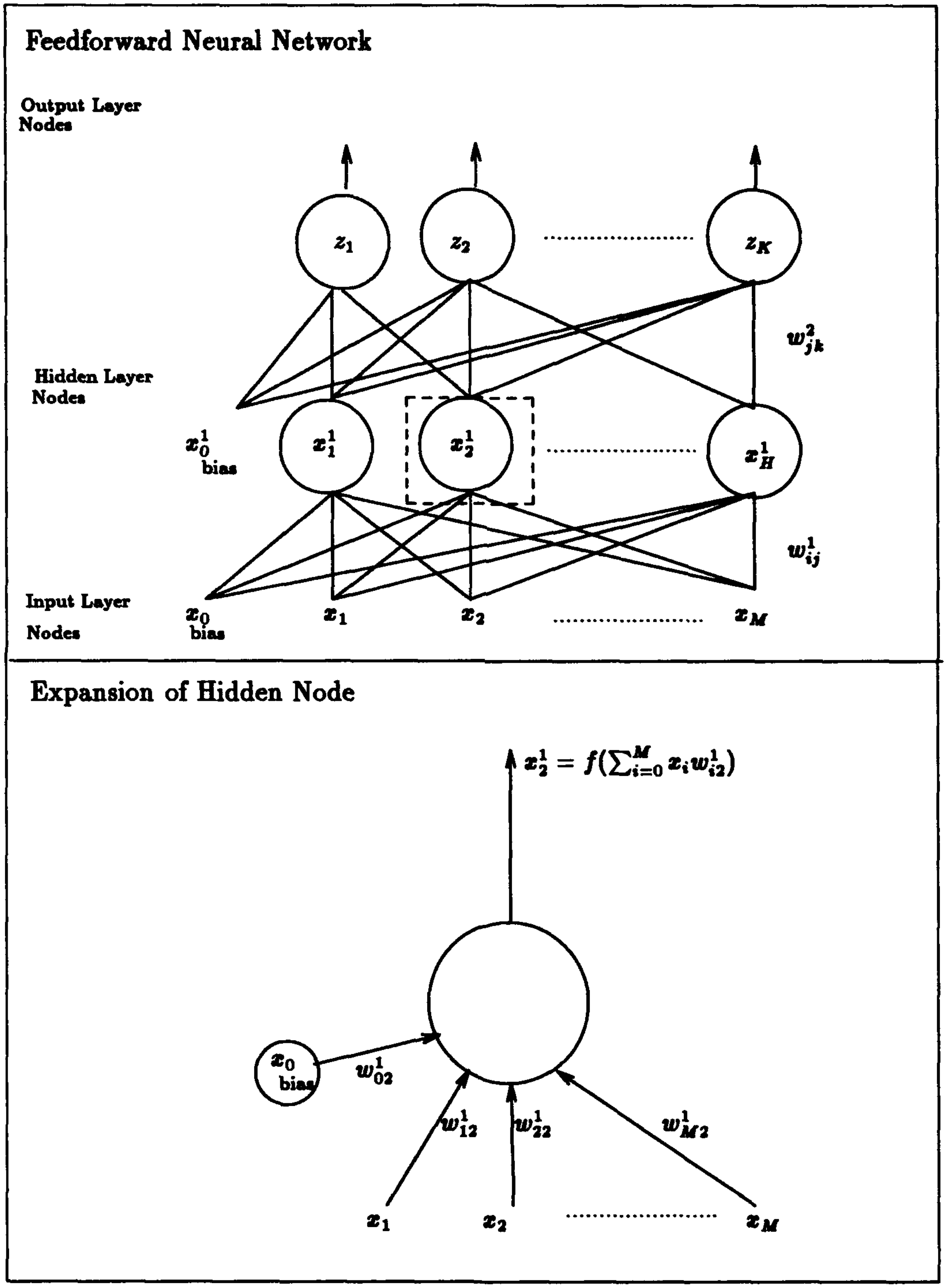
In this section, techniques designed to evaluate and select features within a neural network framework are reviewed. Notational conventions, network structure, and the back propagation learning algorithm are reviewed in Chapter I for neural networks.

*2.5.1 Feature Saliency Metrics.* In this section, the established feedforward neural network feature metrics are described. The single hidden layer neural network is displayed again for reference in Figure 3.

The probability of error measure, $P_e(\mathbf{x})$, introduced in Section 2.4, Equation 16, is often used as a bench mark for independently evaluating the usefulness of neural network features for discriminant analysis problems [51, 59, 60]. $P_e(\mathbf{x})$ is defined again here for reference

$$P_e(\mathbf{x}) = \mathrm{E}_{\mathbf{x}}[1 - \max\{P(C_1|\mathbf{x}), \cdots, P(C_k|\mathbf{x})\}],$$

where $\mathrm{E}\{\cdot\}$ is the expectation function with respect to $\mathbf{x}$, $\max\{\cdot\}$ is the maximization function, and $P(C_k|\mathbf{x})$ is the posterior probability of class $k$. The probability of error $P_e(\mathbf{x})$ is approximated with a feedforward neural network when appropriate conditions are met (discussed in Chapter I Section 3). Therefore, the net's approximation to $P_e(\mathbf{x})$ can also be used to independently evaluate the usefulness of neural network features.


Feedforward Neural Network

Expansion of Hidden Node

Figure 3. Single Hidden Layer Feedforward Neural Network

The network's approximation can be defined as

$$\hat{P}_e(\mathbf{x}, \hat{\mathbf{w}}) = P^{-1} \sum_{p=1}^{P} [1 - \max\{z_1(\mathbf{x}, \hat{\mathbf{w}}), \cdots, z_K(\mathbf{x}, \hat{\mathbf{w}})\}\,,]$$

where $P$ is the number of data exemplars $\mathbf{x}$ in the data set and $z_k(\mathbf{x}, \hat{\mathbf{w}})$ is the network output for the $k$th class. The network approximation to probability of error is computed for the feature vector $\mathbf{x}$ and the estimated network weight parameters $\hat{\mathbf{w}}$. Alternatively, $\hat{P}_e(\mathbf{x}, \hat{\mathbf{w}})$ can be used to evaluate a subset of features with the network. In this case, the correlations between specific feature inputs considered are taken into account.

Le Cun and others propose a saliency metric for evaluating features or middle nodes which is based on second derivative information [39]. They construct a local model of the network error function using a Taylor series approximation of the network's error (see Appendix A for details) Then, the network's change in error due to weight changes is approximated. In order to make it computationally practical to evaluate the Taylor series expression for the change in error, three simplifying assumptions are used:

1. Assume the Taylor series is evaluated at a local minimum of the error which makes the first order terms equal to zero.
2. A diagonalizing assumption is used to eliminate all cross terms of the Hessian matrix.
3. A quadratic approximation assumption about the error surface is used which implies that 3rd order and higher order terms are negligible.

All that remains are the second order diagonal terms, which are assumed positive at a local minimum. Therefore, a perturbation of the vector of estimated weight parameters $\hat{\mathbf{w}}_i^1 = \{\hat{w}_{i1}, \cdots, \hat{w}_{iH}\}$ associated with feature $i$ should cause the error to either increase or to stay the same. For a trained network, the saliency metric for feature node $i$ developed by Le Cun and others uses the second derivative of network output error with respect to the vector of associated feature input weights. The network output error is defined as the squared output error $\mathcal{E}_o$ defined as

$$\mathcal{E}_o = P^{-1}\sum_{p=1}^{P}\sum_{k=1}^{K}(d_k^p - z_k^p)^2$$

The second order saliency metric for feature $i$, denoted $s_i$, is defined as

$$s_i = h_{ii}\frac{x_i^{p2}}{2}$$

where $x_i^p$ denotes the $p$th input exemplar, and

$$h_{ii} = \sum_{j=1}^{H}\frac{\partial^2(\mathcal{E}_o)}{\partial(w_{ij}^1)^2}$$

As defined above, $s_i$ corresponds to the saliency of feature $i$ for a single input exemplar. When the metric $s_i$ is averaged over the entire training set, the result, denoted $\bar{s}_i$, is given as

$$\bar{s}_i = \sum_{p=1}^{P} s_i(x_i^p)$$

where $P$ is the number of training vectors, and $s_i(\mathbf{x}^p)$ indicates that $s_i$ is a function of the $p$th input vector $\mathbf{x}^p$. A more detailed derivation of $s_i$ and $\bar{s}_i$ is shown in Appendix A, Section A.1.

Another technique for measuring the saliency or relevance of a neural network unit, such as a feature, is proposed by Mozer and Smolensky [45]. Mozer and Smolensky propose that the relevance of a feature be measured as a function, $\rho_i$, of how well the network performs with the unit versus how well the network performs without the unit, i.e.

$$\rho_i = \mathcal{E}_{\text{without unit } i} - \mathcal{E}_{\text{with unit } i}$$

They propose approximating $\rho_i$ by examining the derivative of the error with respect to a relevance factor coefficient, $\alpha_i$. The relevance factor coefficient, $\alpha_i$, represents the attentional strength of a unit.

> This coefficient can be thought of as gating the flow of activity from the unit: $o_j = f(\sum_i w_{ji}\alpha_i o_i)$, where $o_j$ is the activity of unit $j$, $w_{ji}$ the connection strength to $j$ from $i$, and $f$ the sigmoid squashing function. If $\alpha_i = 0$, unit $i$ has no influence on the rest of the network; if $\alpha_i = 1$, unit $i$ is a conventional unit. [45:109]

In terms of $\alpha_i$, $\rho_i$ can be defined as

$$\rho_i = \mathcal{E}_{\alpha_i=0} - \mathcal{E}_{\alpha_i=1}$$

and Mozer and Smolensky approximate $\rho_i$ with:

$$\hat{\rho}_i = -\left.\frac{\partial \mathcal{E}}{\partial \alpha_i}\right|_{\alpha_i=1}$$

The derivation and notational details of $\rho_i$ are shown in Appendix A, Section A.2.

Ruck describes a feature saliency metric which measures feature $i$'s effect on a neural network's output [59]. The metric attempts to capture the total of the partial derivatives of the network's outputs with respect to the entire $M$-dimensional feature space $\mathcal{R}^M$. Ruck's saliency metric for feature $i$ is built from the exact partial derivatives of network outputs, $z_k$, with respect to feature inputs $x_i$ using a trained network.

Ideally, the input space would be systematically sampled over its entire range of values [59:34]. If $R$ points were used for each input, the total number of derivatives would be on the order of $R^M$, where $M$ is the number of feature inputs. For other than very small problems, $R^M$ is computationally impractical. Ruck proposes a sampling method which is computationally practical. For every training vector, each feature input is sampled over its range while the other feature inputs are fixed as determined by the actual training vector being evaluated. For $P$ training vectors, the number of derivative evaluations is $KPRM$, where $K$ is the number of output classes represented by the net, M is the number of features, and $R$ is the number of samples for each feature input of each training vector. For the saliency computation of each feature, the set of "pseudo" data points remains the same. Following Reinhart's notation [53:21-22], define $\mathbf{d}_m$ as the vector of $R$ uniformly spaced pseudo points covering the range of the $m$th input feature. The $r$th component, $d_r$, of $\mathbf{d}$ can be defined as:

$$\mathbf{d}_r = \min x_m + (r-1)\frac{\max x_m - \min x_m}{R-1} \qquad r = 1, 2, \cdots, R \tag{17}$$

The Ruck saliency metric for feature $i$, $\hat{\Lambda}_i$, is defined as

$$\hat{\Lambda}_i = \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{r=1}^{R}\sum_{k=1}^{K}\left|\frac{\partial z_k}{\partial x_i}(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})\right| \tag{18}$$

where $P$ is the number of training vectors $\mathbf{x}$; $M$ is the number of features; $R$ is the number of uniformly spaced points covering the range of each input feature found in the training set; $K$ is the number of output classes; the vector $\mathbf{x}^p_{m(r)}$ is the $p$th exemplar $\mathbf{x}^p$ with its $m$th component replaced by, $d_r$ the $r$th component of $\mathbf{d}_m$; and $(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})$ indicates that the derivative is evaluated with the feature vector $\mathbf{x}^p_{m(r)}$ and the final estimates of the trained network weight parameters $\mathbf{w}$. Also, the absolute value of the derivatives are used; therefore, positive and negative derivative changes do not cancel each other. Empirical results indicate that $\hat{\Lambda}_i$ provides similar rankings to $P_e$ [59:46].

Equation 18 has been modified from Ruck's original presentation to reflect that for each vector there are $PRM$ function evaluations of the network's sensitivity $\sum_{k=1}^{K}\left|\frac{\partial z_k}{\partial x_i}(\mathbf{x}, \hat{\mathbf{w}})\right|$ as Ruck intended, rather than $PR$ evaluations as suggested by Ruck's notation [59, 62].

Priddy illustrates a relationship between the class specific Bayesian probability of error and the derivative-based feature saliency metric, $\hat{\Lambda}_i$ [51]. The relationship relies on the assumptions necessary for feedforward neural networks to approximate a Bayes optimal discriminant function [30, 54, 61, 68, 75] [41:50]. When these assumptions are met, the trained feedforward neural network output $z_k$ can be interpreted in the limit as $P(C_k|\mathbf{x})$, which is the posterior probability of class $k$ for $\mathbf{x}$. Since $\sum_{k=1}^{K} P(C_k|\mathbf{x}) = 1$, the neural network approximation to the class specific probability of error for class $k$ is defined as

$$\begin{aligned}\hat{P}_e(k, \mathbf{x}, \hat{\mathbf{w}}) &= 1 - z_k(\mathbf{x}, \hat{\mathbf{w}}) \\ &= \sum_{j \neq k}^{K} z_j(\mathbf{x}, \hat{\mathbf{w}})\end{aligned}$$

Using Ruck's method of feature space sampling, Priddy suggests a Bayesian-based saliency metric, $\Omega_i$. It is defined using the modified notation shown in Equation 18 as [51]:

$$\Omega_i = \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} \left| \frac{\partial \hat{P}_e(k, \mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})}{\partial x_i} \right|, \tag{19}$$

where $P$ is the number of training vectors $\mathbf{x}$; $M$ is the number of features; $R$ is the number of uniformly spaced points covering the range of each input feature found in the training set; $K$ is the number of output classes; the vector $\mathbf{x}^p_{m(r)}$ is the $p$th exemplar $\mathbf{x}^p$ with its $m$th component replaced by, $d_r$ the $r$th component of $\mathbf{d}_m$ defined in Equation 17; and $(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})$ indicates that the derivative is evaluated with the feature vector $\mathbf{x}^p_{m(r)}$ and the final estimates of the trained network weight parameters $\mathbf{w}$.

In neural network terms, $\Omega_i$ is defined as

$$\Omega_i = \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} \left| \sum_{l \neq k}^{K} \frac{\partial z_l(\mathbf{x}^p_{m(r)}, \mathbf{w})}{\partial x_i} \right| \tag{20}$$

Using the triangle inequality, Priddy shows that $\Omega_i$ is bounded above by a simplified saliency metric $\hat{\Omega}_i$, where $\Omega_i \leq \hat{\Omega}_i$ [51]:

$$\hat{\Omega}_i = \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} \sum_{l \neq k}^{K} \left| \frac{\partial z_l(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})}{\partial x_i} \right| \tag{21}$$

Priddy shows $\hat{\Omega}_i$ is a scalar multiple of the $\hat{\Lambda}_i$, i.e.

$$\hat{\Omega}_i = (K - 1)\hat{\Lambda}_i,$$

where $K$ is the number of output classes [51].

Since Priddy's metric is developed from Ruck's metric, Equations 19, 20, and 21 have been modified from Priddy's original presentation. This reflects $PRM$ or $(K-1)PRM$ sensitivity function evaluations (i.e. $\sum_{k=1}^{K} \left|\frac{\partial z_l(\mathbf{x},\hat{\mathbf{w}})}{\partial x_i}\right|$) for each vector similar to Equation 18, rather than $PR$ or $(K-1)PR$ function evaluations as denoted in Priddy's original notation (see discussion on page 51) [51].

Tarr suggests using a weight-based metric, $\Upsilon_i$, for measuring feature $i$'s saliency [72]. This metric depends only on the training exemplars used in training the neural network. The weight saliency is defined as

$$\Upsilon_i = \sum_j (\hat{w}_{ij}^1)^2, \tag{22}$$

where $\hat{w}_{ij}^1$ denotes the estimated weight parameter connecting input feature $i$ to hidden node $j$ [72:45]. The idea is that the weights emanating from important features would grow the most; the weights emanating from less important features would grow less; and the weights emanating from unimportant features would fluctuate up and down about zero. The effectiveness of $\Upsilon_i$ depends on two things [72:45].

1. The weight parameters must be from a trained neural network of the appropriate complexity.
2. The vectors of input features must be normalized to about the same range.

Computationally, this metric is much simpler than the other available feature saliency metrics. Steppe reports the Minkowski$_1$ (taxi-cab) and Minkowski$_\infty$ (infinity) norms of the weights provide similar feature rankings to Tarr's weight saliency which is defined using the squared Minkowski$_2$ norm [70]. Tarr reports that $\Upsilon_i$ provides rankings similar to $\hat{\Lambda}_i$ [72:49].

*2.5.2 Sensitivity Analysis Procedures.* Sensitivity analysis procedures for feedforward neural network feature inputs are related to feature selection. This section reviews sensitivity analysis procedures for neural network feature inputs using exact first and second order partial derivatives [21, 22, 78], and estimated first order partial derivatives [33].

Werbos summarizes a collection of algorithms involving differentiation and cost minimisation [78]. These algorithms are all variations of neural networks using a form of error backpropagation for cost minimization. The first partial derivatives are commonly used in sensitivity analysis. Further information is also provided by the second partial derivatives, $\frac{\partial^2 z_k}{\partial x_i \partial x_j}$. The second partials can be used for understanding the effect of feature interactions on model dynamics [78:766]. Werbos recommends calculating exact first and second order derivatives of the neural network at each data point. According to Werbos, exact partial derivatives are more accurate and easier to compute than attempting to estimate the partial derivatives using input variable perturbation [78:764,766].

Hashem presents mathematical expressions of first and second order neural network feature sensitivities [22]. The expressions are generalized for a neural network with more than one hidden layer which has sigmoidal activation functions on the hidden and output layers. Hashem computes feature sensitivities using exact first and second partial derivatives of $z_k$ with respect to $x_i$, i.e. $\frac{\partial z_k}{\partial x_i}$ and $\frac{\partial^2 z_k}{\partial x_i \partial x_j}$. For approximating a two dimensional function $g(x) = \sin(4x)$ with $x \in [-1, 1]$, Hashem demonstrates how the exact partial derivatives of the neural network correspond closely to the exact partial derivatives of the true function. Hashem does not present methodologies for analyzing or summarizing these sensitivities for studying features.

Guo and Urig studied a neural network model of nuclear power plant thermal performance data to identify the variables which strongly affect the heat rate [21]. In their study, they calculated a feature's sensitivity in a global or average sense. For each feature, the absolute values of the neural network partial derivatives of output $k$ with respect to feature $i$, $\mid \frac{\partial z_k}{\partial x_i} \mid$, are averaged over all known exemplars. This metric differs from Ruck's metric in three ways. First, only the training data are used. Second, it represents an average versus a summation over the feature space. Third, it only examines sensitivities with respect to one output at a time rather than summing over all outputs.

Guo and Urig use their metric to rank order features with respect to their sensitivities for a specific output. They do not suggest improving network performance by selecting a reduced set of the most sensitive variables according to their sensitivity metric. However, they do suggest doing another level of sensitivity analysis to determine the input variables which strongly affect the most sensitive inputs [21:457]. To determine a second level of sensitivity, a neural network is trained to predict the most sensitive feature variable using the remaining feature variables as network inputs. Then the sensitivities are determined as before using Guo and Urig's sensitivity metric for the neural network. Guo and Urig propose that the information about sensitive input variables used by plant personal to determine which efforts will be most effective in improving nuclear power plant efficiency [21].

Klimasauskas investigates the impact of small changes in feature inputs on the neural network output activations using estimated first order partial derivatives [33]. When measuring the sensitivity of feature $i$ on neural network outputs, Klimasauskas "samples" two additional unknown exemplars for an input exemplar which are used for estimating partial derivatives of the neural network model. The original exemplar remains the same except that it is perturbed a small amount above and below the value of feature $i$. For each known exemplar, the partial derivative of $z_k$ with respect to $x_i$ is estimated by taking the difference of the trained neural network's output at the two additional exemplar samples divided by the total change in feature $i$ for the two additional exemplars. Klimasauskas estimates the partial derivatives of the neural network output rather than using exact partial derivatives of the trained neural network output as Werbos does [78]. Using two-dimensional plots, Klimasauskas studies the estimated derivatives over the entire feature space for sensitivity analysis. Two-dimensional plots display the numerical value of feature $i$'s partial derivatives for each input exemplar with the $x$ and $y$ axes being feature $i$'s and $j$'s values [33:22]. For a simple problem, the derivatives are significantly different than zero only at the classification borders where the neural network output transitions from one classification state to another [33:22].

*2.5.3 Selection Methodologies.* Exhaustive enumeration of the feature subsets can be accomplished using either a prediction-type error or a $P_e$-type criterion depending on the neural network application. However, this method becomes impractical in a situation with more than a few variables. It is impractical for two reasons: the large number of enumerated subsets, and the computational requirements of training a neural network.

Practical feature selection methodologies available for neural networks can be divided into three broad categories. The first category selects the $k$ best features using one of the feature saliency metrics described earlier. The second category involves screening a feature set for "noise" features. The third category involves hypothesis testing for the presence of irrelevant features.

The first category of selection methodologies, selecting the $k$ best features, includes several of the feature metrics described in Section 2.5.1. For these metrics, the $k$ best features are selected according to the metric rankings, although no guidance or procedure accompanies the metrics for determining $k$ [39, 59, 51, 72].

In the second category of selection methodologies, Belue and Bauer offer a feature screening technique for identifying a "noise" feature [5]. Their procedure requires adding a noise feature into the original set of features. The neural network is trained with the augmented set of features, and the saliency of all features is computed using either the $\hat{\Lambda}_i$ or $\Upsilon_i$ metric. The training and saliency computation is repeated many times with randomized initial weights in order to characterize the saliency distribution of the noise feature. Belue and Bauer recommend selecting only the $k$ features whose mean saliency falls outside a one-sided confidence interval for the mean saliency of the noise [5].

White's irrelevant input hypothesis test (for neural networks trained with backpropagation) falls into the third category of feature selection methodologies [82]. The hypothesis testing methodology requires the computation of a chi-squared test statistic to identify when a vector of weights connected to an feature input are irrelevant (i.e. can not be rejected as statistically different than zero). White's hypothesis test is expressed as

$$H_0 : \mathbf{Sw}^* = \mathbf{0},$$

where S is a $q \times s$ selection matrix picking out the $q$ elements of the $s \times 1$ vector of neural network optimal weights $\mathbf{w}^*$ which are hypothesized to be zero under $H_0$. When $H_0$ is true, the $q$ elements of the estimated weight vector $\hat{\mathbf{w}}$, selected by $\mathbf{S}\hat{\mathbf{w}}$, are typically weights which are small in absolute magnitude.

White's irrelevant input hypothesis test requires that the limiting distribution of $\hat{\mathbf{w}}$ as it converges to $\mathbf{w}^*$ is a multivariate normal distribution [79]. The limiting distribution of $\hat{\mathbf{w}}$ will be multivariate normally distributed in the limit if the redundant inputs and/or irrelevant hidden units are removed [79:441]. When $\hat{\mathbf{w}}$ has a multivariate normal limiting distribution, then

$$\sqrt{P}(\hat{\mathbf{w}} - \mathbf{w}^*) \sim N_s(\mathbf{0}, \mathbf{C}^*),$$

where $P$ is the number of data exemplars [79]. White shows that the multivariate normal distribution of $\sqrt{P}(\hat{\mathbf{w}} - \mathbf{w}^*)$ implies the following is true [79]:

$$\sqrt{P}\mathbf{S}(\hat{\mathbf{w}} - \mathbf{w}^*) \sim N_q(\mathbf{0}, \mathbf{SC}^*\mathbf{S}')$$

When $H_0$ is true, then $\mathbf{Sw}^* = \mathbf{0}$ [79]. This implies

$$\sqrt{P}\mathbf{S}\hat{\mathbf{w}} \sim N_q(\mathbf{0}, \mathbf{SC}^*\mathbf{S}'),$$

and therefore

$$P\hat{\mathbf{w}}'\mathbf{S}'(\mathbf{SC}^*\mathbf{S}')^{-1}\mathbf{S}\hat{\mathbf{w}} \sim \chi^2_q \qquad (23)$$

An analytical expression for $\mathbf{C}^*$ is not available, but an estimator $\hat{\mathbf{C}}$ exists which is weakly consistent, where

$$\hat{\mathbf{C}} = \hat{\mathbf{A}}^{-1}\hat{\mathbf{B}}\hat{\mathbf{A}}^{-1} \tag{24}$$

The matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ are defined

$$\begin{aligned}\hat{\mathbf{A}} &= P^{-1}\sum_{p=1}^{P} \nabla^2 \mathcal{E}(\mathbf{x}^p, \hat{\mathbf{w}}) \\ \hat{\mathbf{B}} &= P^{-1}\sum_{p=1}^{P} \nabla \mathcal{E}(\mathbf{x}^p, \hat{\mathbf{w}}) \nabla \mathcal{E}(\mathbf{x}^p, \hat{\mathbf{w}})'\end{aligned}$$

where $\mathcal{E}$ is the neural network error used for training, and the operators $\nabla$ and $\nabla^2$ denote the $s \times 1$ gradient and the $s \times s$ Hessian operators of $\mathcal{E}$ defined with respect to $\hat{\mathbf{w}}$, where $\mathbf{x}^p$ is the $p$th input exemplar, $x^p$. Replacing $\mathbf{C}^*$ with $\hat{\mathbf{C}}$ does not affect the limiting $\chi_q^2$ distribution. However, White warns that sometimes a much larger sample size $P$ is required to obtain a good approximation of $\mathbf{C}^*$ [79:442-443]. The $\chi_q^2$ test statistic defined in Equation 23 is used to test $H_0$ at a desired accuracy of $1 - \alpha$. Whenever, the test statistic exceeds the $1 - \alpha$ percentile of $\chi_q^2$, the irrelevant input hypothesis is rejected. The probability of failing to reject $H_0$ when $H_0$ is false is equal to $\alpha$.

### *2.6 Summary*

Feature evaluation metrics and feature selection techniques developed for linear regression, nonlinear regression, discriminant analysis, and feedforward neural networks are surveyed in the body of this chapter. In the remainder of the dissertation, research results are presented which in some cases were derived from the body of knowledge surveyed in this chapter.

In Chapter III, improved neural network feature saliency metrics are introduced along with a catalogue of all the available saliency metric definitions and relationships. A good number of these metrics are surveyed in Section 2.5. The saliency screening technique and the irrelevant input hypothesis test reviewed in Section 2.5 are used for the results presented in Chapter IV. To determine a practical model selection criteria for neural network selection, the linear and nonlinear regression model se'ection criteria from Sections 2.2 and 2.3 are studied. The sequential selection algorithms from linear regression and discriminant analysis are the basis for the backwards sequential algorithm used in the procedures developed in Chapters V and VI.

# *III. Feedforward Neural Network Feature Saliency Metrics*

## *3.1 Introduction*

Feature saliency metrics are used for evaluating and ranking individual features within a neural network. In this research, the terminology metric is used to refer to a measure of feature importance. The results shown in this chapter are important for three reasons. First, new and improved feature saliency metrics are presented. Second, saliency metric sensitivities to sampling, training, and redundant middles nodes are documented. Third, a catalogue of feature saliency metric definitions and interrelationships is presented which consolidates the set of available neural network feature saliency metrics. An overview of this chapter follows.

In Section 2 of this chapter, a framework for understanding derivative-based saliency is discussed. Several variations of derivative-based saliency are investigated, including a known data metric which requires fewer derivative evaluations than an established metric. The saliency metrics are evaluated for their sensitivities to sampling, training, and redundant middle nodes. In Section 3, a mathematical relationship is derived between derivative-based saliency and the weight-based saliency. $P_e$-based feature saliency metrics and related research results are discussed in Section 4. These results include the illustration of a precise relationship between an established $P_e$-based metric and an established derivative-based metric, the introduction of a new $P_e$-based feature saliency metric, and derivation of relationships between the new $P_e$-based metric and the improved derivative-based metric. In Section 5, a catalogue of definitions and theoretical relationships among the set of available feature saliency metrics is presented. Also in Section 5, feature saliency results are documented on a 'real world' problem for the set of available feature metrics. The results presented in this chapter are summarized in Section 6.

The research and theoretical results presented in this chapter reflect the exclusive use of the sigmoidal activation functions on the middle and output nodes of a feedforward neural network (presented in Chapter I). The fundamental network output and network derivative definitions will change for other types of activation functions. However, similar the underlying concepts and relationships shown throughout this chapter will still hold. The neural network notational conventions, network structure, and backpropagation algorithm introduced in Section 3 of Chapter I, as well as the feature saliency notation reviewed in Section 4 of Chapter II, are used as necessary in this chapter.

### *3.2 Derivative-based Feature Saliency Metrics*

In this section, neural network partial derivatives and related notations are reviewed. Then, an integrated saliency metric is introduced as a 'truth model' for derivative-based saliency. Several approximations for integrated saliency are discussed since the integrated saliency metric is intractable for most problems. The first approximation is an established saliency metric proposed by Ruck which involves evaluating the saliency with what could be called 'pseudo-samples' from the feature space [60]. The second approximation is similar to Ruck's, but the saliency is evaluated with random samples from the feature space. The third approximation is also similar to Ruck's, but the saliency is evaluated with only the known data from the feature space. All of these derivative-based saliency metrics are analyzed for their sensitivity to sampling, training length, and redundant middle nodes. Finally, a summary of the derivative-based saliency results is presented.

*3.2.1 Background.* The importance of an input feature is a function of the network's sensitivity to changes in the input feature [21, 22, 33, 59, 78]. Evaluating the network's sensitivity to the $i$th feature input is analogous to evaluating partial derivatives of the network output with respect to the $i$th feature input.

A slight digression is necessary to review some of the neural network notation related to evaluating partial derivatives. Let the input features $x_i$ be indexed from $i = 0, \cdots, M$, the middle node activations $x_j^1$ be indexed from $j = 0, \cdots, H$, and the [illegible]t node activations be indexed from

$k = 1, \cdots, K$. Let $f(a)$ represent the sigmoidal nonlinear activation function defined as:

$$f(a) = \frac{1}{1 + e^{-(a)}}$$

When sigmoidal activation units are used on the middle and output nodes, the feedforward neural network output and hidden node activations and associated specialized terms are defined as follows:

$$\begin{aligned} z_k &= f_h(\sum_{j=1}^{H} \hat{w}_{jk}^2 x_j^1) \\ \delta_k^2 &= z_k(1 - z_k) \\ x_j^1 &= f_h(\sum_{i=1}^{M} \hat{w}_{ij}^1 x_i) \\ \delta_j^1 &= x_j^1(1 - x_j^1), \end{aligned}$$

where $x_0$ and $x_0^1$ are bias terms which are equal to one, $\hat{w}_{jk}^2$ is an estimate of the weight parameter connecting the $j$th middle node with the $k$th output, $\hat{w}_{ij}^1$ estimates the weight parameter connecting the $i$th feature input with the $j$th middle node. Now, applying partial differentiation to $z_k$ with respect to $x_i$ gives:

$$\frac{\partial z_k}{\partial x_i} = \delta_k^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jk}^2 \hat{w}_{ij}^1 \tag{25}$$

The definitions for $z_k$, $x_j^1$, and $\frac{\partial z_k}{\partial x_i}$ are significantly different when sigmoidal activation functions are not used.

In Figure 4, there are three examples of classification problems displayed. These three examples encompass the range from output classes not overlapping (Example 1) to output classes significantly overlapping (Example 3). For each example shown in Figure 4, a neural network was trained on 200 training vectors using two output nodes and one middle node with a step size of 0.3 and a momentum of 0.7. Training was discontinued when the training-test set error was minimized. This occurred at two, five, and ten epochs for the three examples.

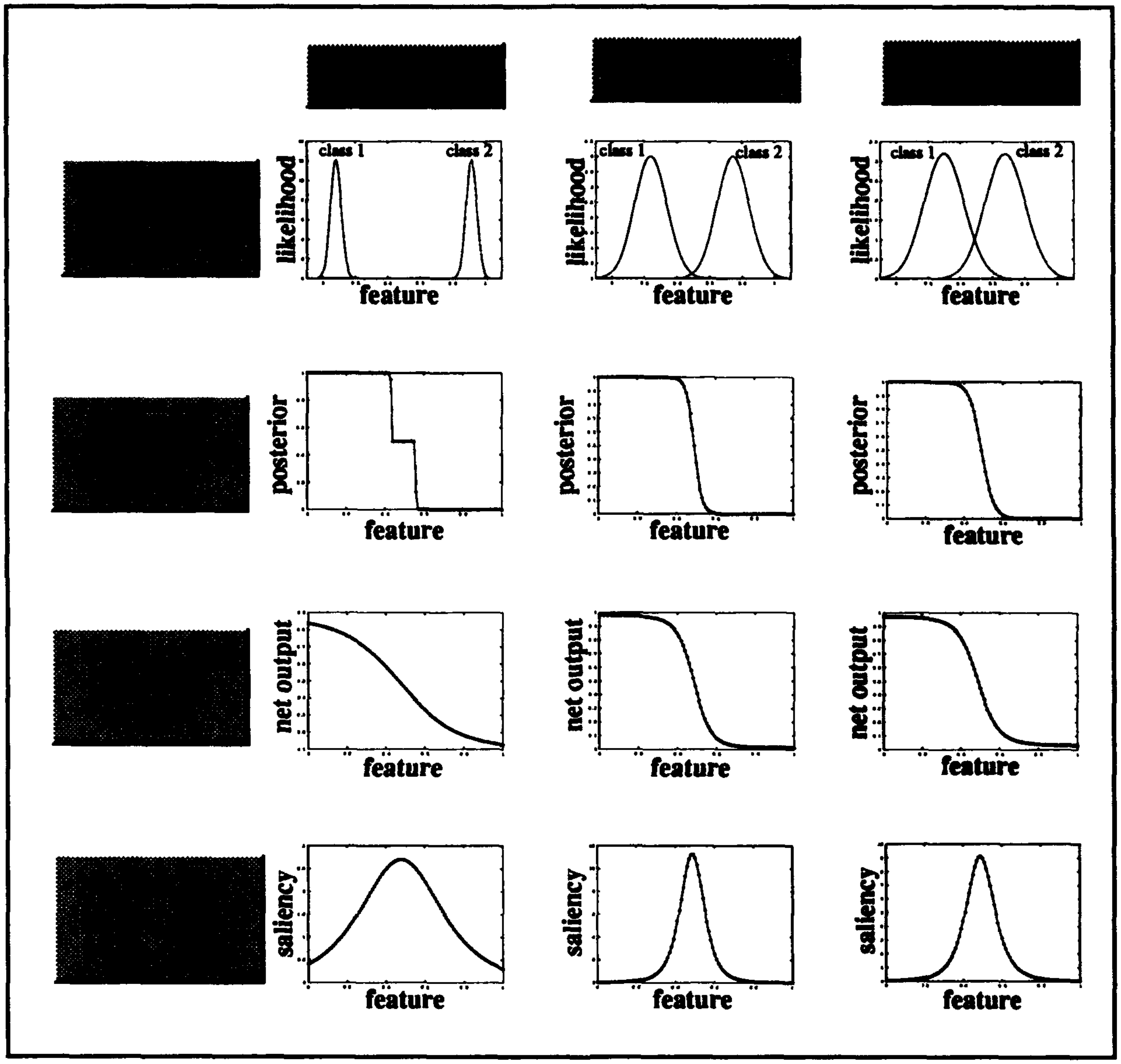


Figure 4. Three Examples of a Two-Class Univariate Normal Problem

In the first row of Figure 4, the true likelihood function of the data in each class is shown. For all three examples, the true underlying distribution function is the normal distribution function, denoted $h(\mathbf{x})$, which is

$$h(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

where $\mu$ and $\sigma$ are the expected value and standard deviation of $x$.

In the second row of Figure 4, the *a posterior* distribution function of $\mathbf{x}$ is shown. The *a posterior* distribution function for the $k$th class, denoted $P(C_k|\mathbf{x})$, is defined from Bayes rule as

$$P(C_k|\mathbf{x}) = \frac{P(C_k)h_k(\mathbf{x})}{\sum_{j=1}^{K} P(C_j)h_j(\mathbf{x})},$$

where $P(C_j)$ denotes the prior probability of class $k$ and $h_k(\mathbf{x})$ denotes the likelihood function for class $k$.

In the third row of Figure 4, the neural network's output function for class one is shown. In this network, the outputs from each class can be interpreted as an approximation to the *a posterior* distribution for $\mathbf{x}$ (see discussion Section 3 of Chapter 1). Notice in Example 1, the neural network output for class one is a poor approximation to the *a posterior* distribution when the classes are not overlapping. However, when the tails of the two classes do overlap in Examples 2 and 3, the neural network outputs are better approximations to the *a posterior* probabilities.

In the fourth row of Figure 4, the absolute value of the neural network's feature 'sensitivity function' is shown. For these univariate two-class examples, the 'saliency function' is

$$\sum_{k=1}^{2} \left|\frac{\partial z_k(\mathbf{x}, \hat{\mathbf{w}})}{\partial x_1}\right|,$$

where $z_k(\mathbf{x}, \hat{\mathbf{w}})$ indicates that $z_k$ is evaluated with the univariate feature vector $\mathbf{x}$ and the vector of estimated weight parameters $\hat{\mathbf{w}}$. Notice, the neural network's maximum feature sensitivity corresponds to the classification borders where the neural network's output is equal to $\frac{1}{2}$. For the second and third examples, the region where the network is most sensitive to the features is also the region where the true *a posterior* distribution is most sensitive. However, in Example one where the likelihood distributions are not overlapping, the most sensitive regions of the feature saliency and the *a posterior* distribution do not correspond.

*3.2.2 'Truth Model:' Integrated Feature Saliency.* A comprehensive measure of derivative-based feature saliency is defined as the expected feature sensitivity integrated over the entire feature space. For the univariate discrimination problems shown in Figure 4, this metric entails integrating under the 'saliency function' curves shown in the fourth row. Although the results shown in this section use notation and definitions specific to feedforward neural networks with sigmoid activation functions, this framework can also be applied to neural networks defined with other types of activation functions.

For the second and third examples shown in Figure 4, the region of integration is representative of where the true data exists. However, for the first example, the region of integration includes a portion of the curve where the likelihood of being in either class is zero. This portion of the curve corresponds to large values of the 'saliency function,' yet poor approximations to the *a posterior* distribution.

Define integrated feature saliency as the average value of the saliency function $f_i(\mathbf{x}, \hat{\mathbf{w}}) = \sum_{k=1}^{K} \left| \frac{\partial z_k}{\partial x_i} \right|$ over the feature space region [67:179]. Let the integrated feature saliency, denoted $\Lambda_i$, be given as

$$\Lambda_i = V_i^{-1} \int_{\mathcal{R}^M} \int_{f_i(\mathbf{x},\hat{\mathbf{w}})=0}^{f_i(\mathbf{x},\hat{\mathbf{w}})=\sum_{k=1}^{K} \left| \frac{\partial z_k}{\partial x_i}(\mathbf{x},\hat{\mathbf{w}}) \right|} f_i \, df_i \, d\nabla^M, \tag{26}$$

where $\mathcal{R}^M$ represents the $M$-dimensional feature space region (i.e. $\int_{\mathcal{R}^M} \equiv \int_{x_1} \int_{x_2} \cdots \int_{x_M}$ and $d\nabla^M \equiv d_{x_M} \, d_{x_{M-1}} \cdots d_{x_1}$), and $V_i$ represents the total saliency 'volume' which is given as [67:187]

$$V_i = \int_{\mathcal{R}^M} \int_{f_i(\mathbf{x},\hat{\mathbf{w}})=0}^{f_i(\mathbf{x},\hat{\mathbf{w}})=\sum_{k=1}^{K} \left| \frac{\partial z_k}{\partial x_i}(\mathbf{x},\hat{\mathbf{w}}) \right|} df_i \, d\mathcal{R}^M, \tag{27}$$

The limits of integration on each feature correspond to the observed range of the data inputs. The absolute value of the derivatives are measured to ensure that the positive and negative derivatives do not cancel each other.

A number of numerical methods can be used to evaluate Equation 26. These methods are good approximations for smooth regions with no pockets of highly peaked regions [50:]. With most numerical integration methods, a discrete number of function evaluations are required which is dependent on the number of features $M$ and the number of evaluation points $R$ for each feature dimension. The number of function evaluations increases exponentially with $M$. This means that on the order of $R^M$ function evaluations will be needed. For example, a 10 point Gauss-Legendre integration requires $10^M$ function evaluations for $M$ features. The number of function evaluations is reasonable when $M$ is small, but consider a case where 10 point Gauss-Legendre integration is used for $M = 10$ features. In this case, $10^{10}$ or 10 billion function evaluations of $f_i$ are needed to compute the saliency of each feature.

Clearly, a computationally tractable method is needed for evaluating feature saliency. In the next three sections, tractable methods for approximating $\Lambda_i$ are defined. These approximations are similar, but each one uses a different set of data for evaluating the function $f_i(\mathbf{x}, \hat{\mathbf{w}}) = \sum_{k=1}^{K} \left| \frac{\partial z_k}{\partial x_i}(\mathbf{x}, \hat{\mathbf{w}}) \right|$.

*3.2.3 'Pseudo-Data' Approximation.* The first method for approximating $\Lambda_i$ was proposed (implicitly) by Ruck [59]. This metric is reviewed in Section 2.5 of Chapter II. Ruck's metric involves what could be called 'pseudo-sampling' from the $M$-dimensional feature space $\mathcal{R}^M$.

A description of 'pseudo-sampling' follows [59]. For every training vector, each feature input is sampled uniformly over its observed range while the other feature inputs correspond to the training vector being sampled. This corresponds to $PRM$ 'pseudo-samples,' where $P$ is the number of vectors in the training set, $R$ is the number of uniformly spaced sample points per feature dimension, and $M$ is the number of features as before. For the saliency computation of each feature, the set of "pseudo" data points remains the same. This approximation is computationally tractable, because the number of function evaluations now increase linearly with $M$ rather than exponentially.

For the $i$th feature, Ruck's metric (defined earlier in Equation 18) is described again for reference. Following Reinhart's notation [53:21-22], let $\mathbf{d}_m$ be the vector of $R$ uniformly spaced pseudo points covering the range of the $m$th input feature. The $r$th component, $\mathbf{d}_r$, of $\mathbf{d}$ can be defined as:

$$\mathbf{d}_r = \min x_m + (r-1)\frac{\max x_m - \min x_m}{R-1} \qquad r = 1, 2, \cdots, R \tag{28}$$

The Ruck saliency metric for feature $i$, $\hat{\Lambda}_i$, is defined again for convenience as

$$\hat{\Lambda}_i = \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{r=1}^{R}\sum_{k=1}^{K}\left|\frac{\partial z_k}{\partial x_i}(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})\right| \tag{29}$$

where $P$ is the number of training vectors $\mathbf{x}$; $M$ is the number of features; $R$ is the number of uniformly spaced points covering the range of each input feature found in the training set; $K$ is the number of output classes; the vector $\mathbf{x}^p_{m(r)}$ is the vector $\mathbf{x}^p$ with its $m$th component replaced by, $\mathbf{d}_r$ the $r$th component of $\mathbf{d}_m$; and $(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})$ indicates that the derivative is evaluated with the feature vector $\mathbf{x}^p_{m(r)}$ and the final estimates of the trained network weight parameters $\mathbf{w}$.

The approximation $\hat{\Lambda}_i$ represents the total network saliency for $PMR$ 'pseudo-sampled' data points. Let the average 'pseudo-saliency' be defined

$$\hat{\Lambda}_i^{\text{pseudo}} = (PRM)^{-1}\hat{\Lambda}_i \tag{30}$$

For making empirical comparisons between $\Lambda_i$ and the various approximations to $\Lambda_i$, the average 'pseudo saliency' $\hat{\Lambda}_i^{\text{pseudo}}$ is most appropriate .

*3.2.4 Random Data Approximation.* A second method for approximating $\Lambda_i$ can be defined using random samples from the $M$-dimensional feature space $\mathcal{R}^M$. This approximation is similar to $\hat{\Lambda}_i^{\text{pseudo}}$, but random samples are used instead of 'pseudo-samples.' For data normalized to a unit hypercube, the $n$th random sample is created by drawing a UNF(0, 1) random number for each feature.

The random data approximation, denoted $\hat{\Lambda}_i^{\text{random}}$, is given as

$$\hat{\Lambda}_i^{\text{random}} = (N^r)^{-1} \sum_{n=1}^{N^r} \sum_{k=1}^{K} \left| \frac{\partial z_k}{\partial x_i}(\mathbf{x}^n, \hat{\mathbf{w}}) \right|$$

where $\mathbf{x}^n$ denotes the $n$th random samples drawn from $\mathcal{R}^M$, and $N^r$ is the total number of random samples. The random data approximation to $\Lambda_i$ represents the average network saliency over $N^r$ randomly sampled data points.

*3.2.5 Known Data Approximation.* A third method for approximating $\Lambda_i$ is defined by sampling only the known data. Guo and Urig suggested a similar metric for sensitivity analysis of nuclear power plant thermal data [21]. Guo and Urig's metric is different because they consider the sensitivity for each of the $K$ outputs separately. For the known data saliency, the network's feature sensitivity is evaluated in a manner proportional to the total likelihood function of the data. Note that regions of maximum total likelihood may not correspond to regions of maximum feature saliency, as in Example one in Figure 4 on page 63.

The known data saliency, denoted $\hat{\Lambda}_i^{\text{data}}$, is given as

$$\hat{\Lambda}_i^{\text{data}} = P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} \left| \frac{\partial z_k}{\partial x_i}(\mathbf{x}^p, \hat{\mathbf{w}}) \right| \tag{31}$$

This metric requires a factor of $RM$ fewer computations than $\hat{\Lambda}_i^{\text{pseudo}}$. The known data approximation to $\Lambda_i$ represents the average network saliency over the $P$ known data points.

The various approximations to $\Lambda_i$ are analyzed in the next section. The known data approximation provides results similar to the other metrics. It is probably the best choice for practical use since it evaluates regions of the feature space where the data is known, and it requires fewer calculations than the integrated or pseudo metrics.

*3.2.6 Analysis.* The derivative-based metrics introduced in the previous sections are analyzed in this section. To do this, the three examples shown in Figure 4 are revisited. However, for this analysis, a N(0, 1) random variable, denoted $x_{\text{noise}}$, is added as a second feature to each class. Both features are normalized between 0 and 1. The relative importance of the 'truly salient' feature to noise is analyzed. The saliencies for these examples are evaluated for sensitivities to over-training and redundant middle nodes. The metrics $\hat{\Lambda}_i^{\text{pseudo}}$ and $\hat{\Lambda}_i^{\text{random}}$ are also analyzed for their sensitivity to sampling.

Since these examples are linearly separable, a minimal network (i.e. no redundant middle nodes) consists of one middle node. For each level of training, the minimal network's results are used as the baseline for comparison. Figure 5 is a collection of three dimensional plots summarizing the three examples. Figure 6 and Table 5 are used to summarize the sensitivities of the derivative-based saliency metrics. In Table 6, a general summary of these sensitivities is presented.

For all the examples, the networks used 400 training vectors, two output nodes, and a log-linear declining learning rate (see Chapter I for definition). The neural network plots for each example in Figure 5 are from a single realization of a trained neural network using one middle node. For all of the plots, the $x$-axis corresponds to the variable $x_1$, and the $y$-axis corresponds to the variable $x_{\text{noise}}$.

In the first and second rows of Figure 5, the $z$-axis represents the value of the individual likelihood functions and the trained neural network output functions for each class. In the third and fourth rows, the $z$-axis represents the value of the $i$th 'saliency function' $\sum_{k=1}^{K} |\frac{\partial z_k(\mathbf{x}, \hat{\mathbf{w}})}{\partial x_i}|$. The terminology 'saliency function' is appropriate, because each of the derivative-based saliency metrics are defined as a series of 'saliency function' measurements. In the last row of Figure 5, the $z$-axis represents a ratio of the 'saliency function' for $x_1$ divided by the 'saliency function' for $x_{\text{noise}}$.

The 'saliency functions' shown in the third and the fourth row vary greatly across the feature space. They are most peaked where the neural network's output function has the greatest slope. All three derivative-based metrics obtain markedly different values due to different 'saliency function' measurements. For instance, in the first example, the known data saliency metric would be the smallest, because the known data is from a region where the 'saliency function' is not peaked.

The 'saliency function' ratio shown in the fifth row is a relatively flat or a constant function. Where the ratio is not flat, it fluctuates due to the division of two very small numbers. Also, in the regions where the ratio is not constant, there is very little, if any, true data.

The 'saliency function' ratio at any point can be interpreted as the relative importance of the feature $x_1$ to the feature $x_{\text{noise}}$. The 'saliency function' ratios in Figure 5 indicate that the relative importance of one feature to another is nearly constant regardless of how or where the 'saliency function' is measured. Therefore, when measuring the relative importance of a feature, all of the metrics perform about the same. This can be seen in Figure 6 when comparing integrated and known-data saliencies, and in the last row of Table 5 when comparing the sample saliency function ratios.

The results shown in Figure 6 document the saliency metric sensitivities to training and redundant middle nodes. For completeness, the results documented in Figure 6 are summarized in Table 5. In Figure 6, the $x$-axis correspo[illegible] to the number of middle nodes used, and the $y$-axis corresponds to the value of the 'saliency function' ratio. Each line of the plots corresponds to a different amount of training. The smallest amount of training corresponds to the point where the network classification error is initially minimized. A point on the plots represents an experiment involving 30 neural network runs with the corresponding amount of training and number of middle nodes.

For each experiment, the 'saliency function' ratios for the integrated and the known feature saliency metrics are about the same. The 'saliency function' ratios corresponding to the metrics

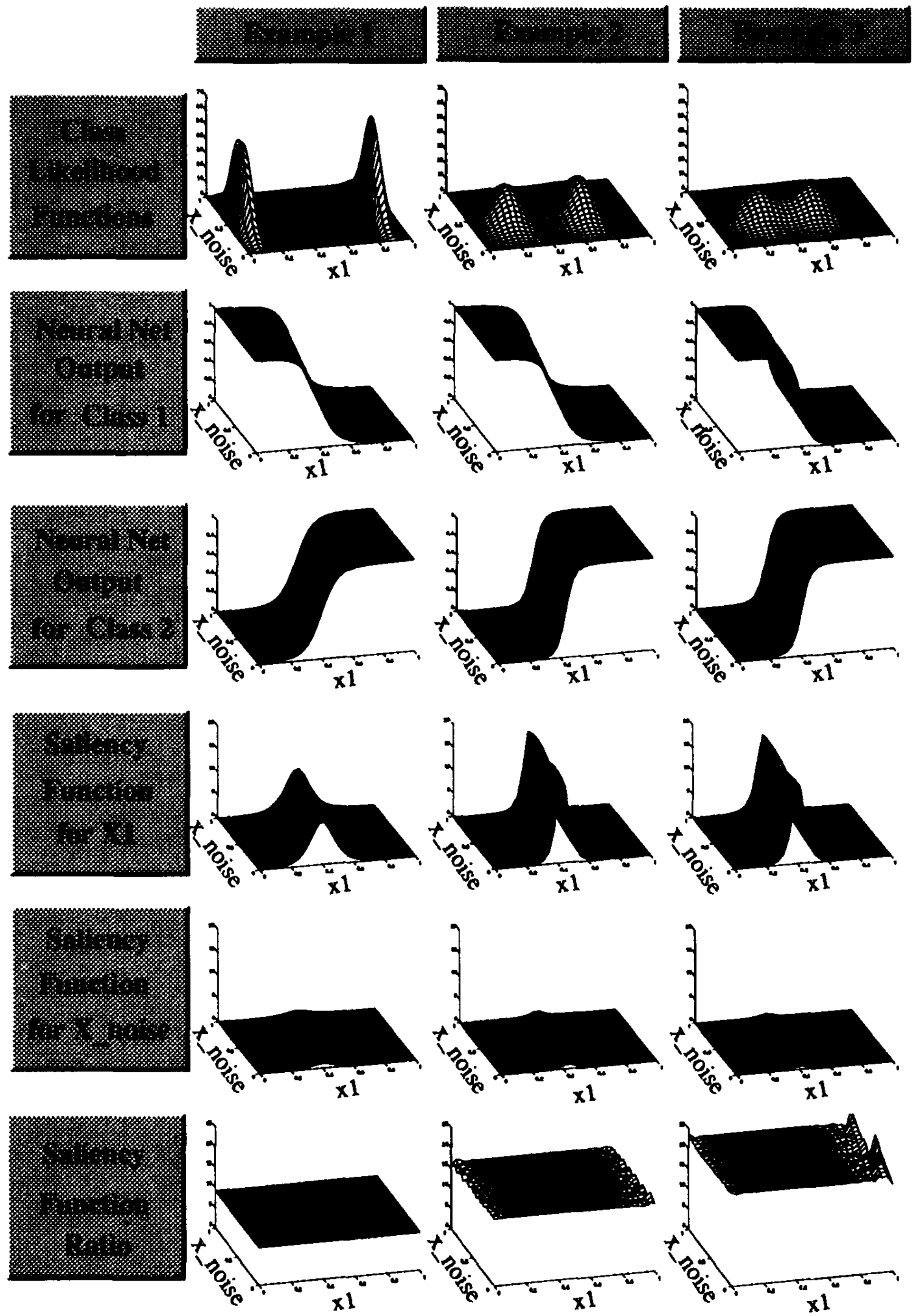


Figure 5. Three Examples of a Two-Class Multivariate Problem

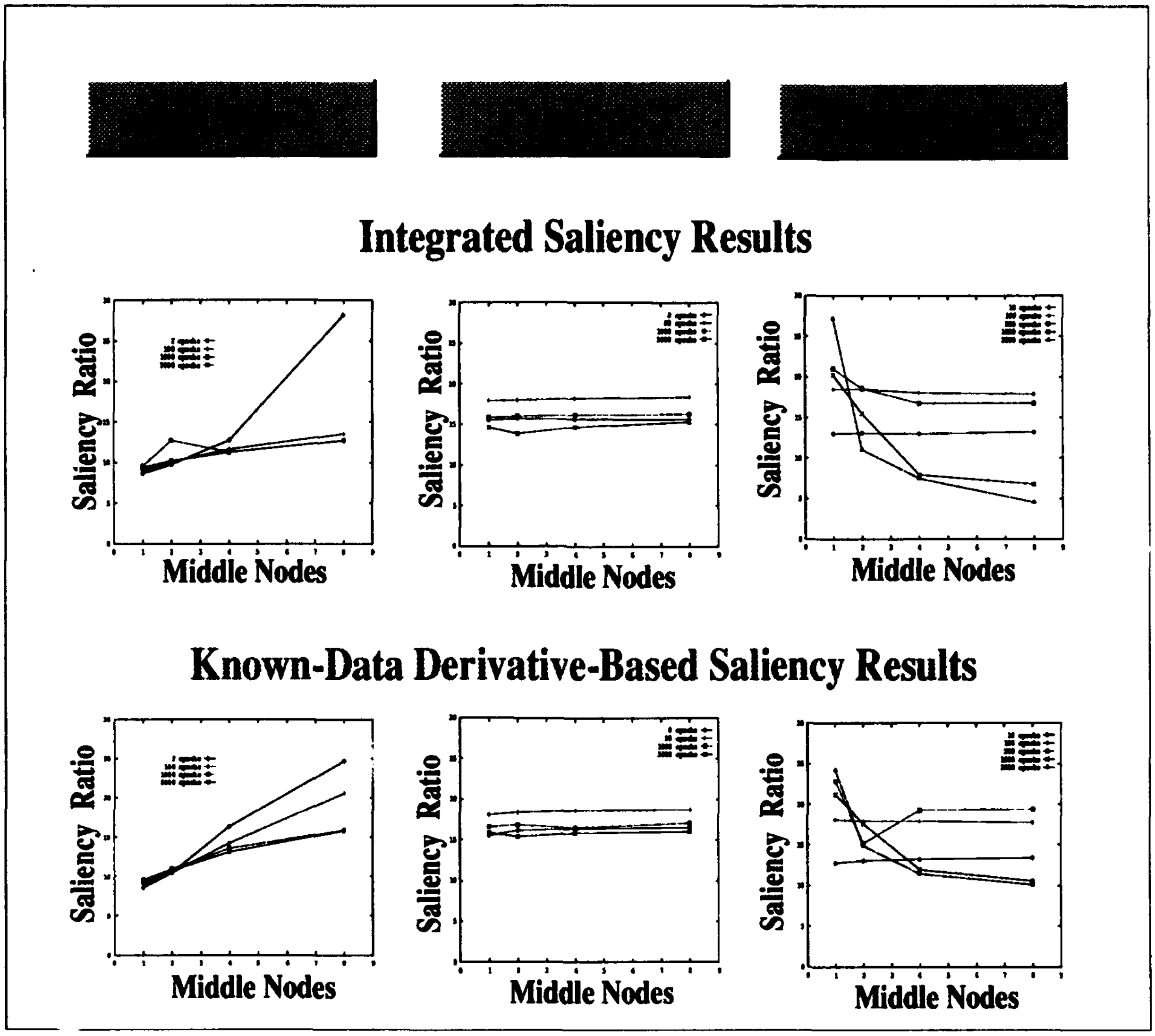


Figure 6. Summary of Middle Node and Training Sensitivities

Table 5. Derivative-Based Saliency Metrics for Three Examples

| | Example 1<br>(tails not overlapping) | Example 2<br>(tails slightly overlapping) | Example3<br>(tails overlapping) |
|---|---|---|---|
| Relative 'saliency function' ratio for 1 middle node | small | large | large |
| Sensitivity to $R$ | none | none | none |
| Sensitivity to increased training for 1 middle node | Saliency Ratio Increases | none | Saliency Ratio Increases |
| Sensitivity to increased training for > 1 middle node | 'saliency function' ratio decreases | 'saliency function' ratio decreases | 'saliency function' ratio decreases |
| Sensitivity to Redundant middle nodes | 'saliency function' ratio increases | 'saliency function' ratio decreases at high amounts of training | 'saliency function' ratio decreases |
| Average 'saliency function' ratios over 30 runs of trained neural networks using 1 middle node | | | |
| Epochs | 100 | 80 | 100 |
| $\Lambda_i$<br>$\hat{\Lambda}_i^{pseudo}$<br>$\hat{\Lambda}_i^{random}$<br>$\hat{\Lambda}_i^{data}$ | 8.991<br>8.985<br>8.985<br>8.988 | 17.906<br>18.148<br>18.147<br>18.158 | 18.48<br>18.03<br>17.96<br>18.11 |

$\hat{\Lambda}_i^{pseudo}$ and $\hat{\Lambda}_i^{random}$ are not shown, since their results do not differ significantly from those shown for the integrated and known saliency metrics in Figure 6.

The first example with non-overlapping classes has the smallest 'saliency function' ratios, in general, and the third example with significantly overlapping classes has the largest 'saliency function' ratios. This was specifically illustrated for one middle node and 100 epochs in Figure 5. For a minimal network of one middle node, the 'saliency function' ratio varies for different amounts of training. In the third example where the two classes are significantly overlapping, there is a definite relationship between training and the 'saliency function' ratio for a minimal network. In this example, the 'saliency function' ratio increases when the network is trained longer. This relationship does not hold for a network trained with redundant middle nodes.

In the presence of redundant middle nodes, a negative relationship exists between the 'saliency function' ratio and additional training. With additional training, the redundant middle nodes begin to incorporate unnecessary information from $x_{\text{noise}}$ without affecting the network error rate. As a result, the saliency of $x_{\text{noise}}$ increases, which in turn deflates the 'saliency function' ratio. This type of behavior is evidence of over-training.

In the first example where the two classes are greatly separated, there is a great deal of flexibility in the function which can effectively discriminate between the two classes. The minimal network trains to a function which is not very steep in slope; however, a network output function with a different slope would also be effective. On the average, a 'saliency function' ratio of about 10 is produced by the minimal network containing one middle node. Interestingly, redundant middle nodes afford the flexibility for a different network output function, so there is an increase in the 'saliency function' ratio when additional middle nodes are added.

For the first example, a network with eight middle nodes produces a distorted output function compared to a network with one middle node. The saliency functions shown in Figure 7 are for one and eight middle nodes after five epochs of training. At eight middle nodes, there is graphical evidence that the noise feature is affecting the neural network saliency functions. After about five epochs, the middle nodes begin 'training' to the information in $x_{\text{noise}}$ which deflates the 'saliency function' ratio. Inspection of the weights associated with $x_{\text{noise}}$ confirms that the proportional influence of $x_{\text{noise}}$ grows with increased training.

For the second and third examples where the two classes are overlapping, the minimal network trains to a function which is very steep. There is not as much flexibility in the 'choice' of a network output function which can effectively separate the classes. For these examples, the 'saliency function' ratio does not increase in the presence of redundant middle nodes. After sufficient training, the redundant middle nodes begin to incorporate unnecessary information from $x_{\text{noise}}$, which makes the saliency function between $x_1$ and $x_{\text{noise}}$ decrease. This behavior is best seen when looking at

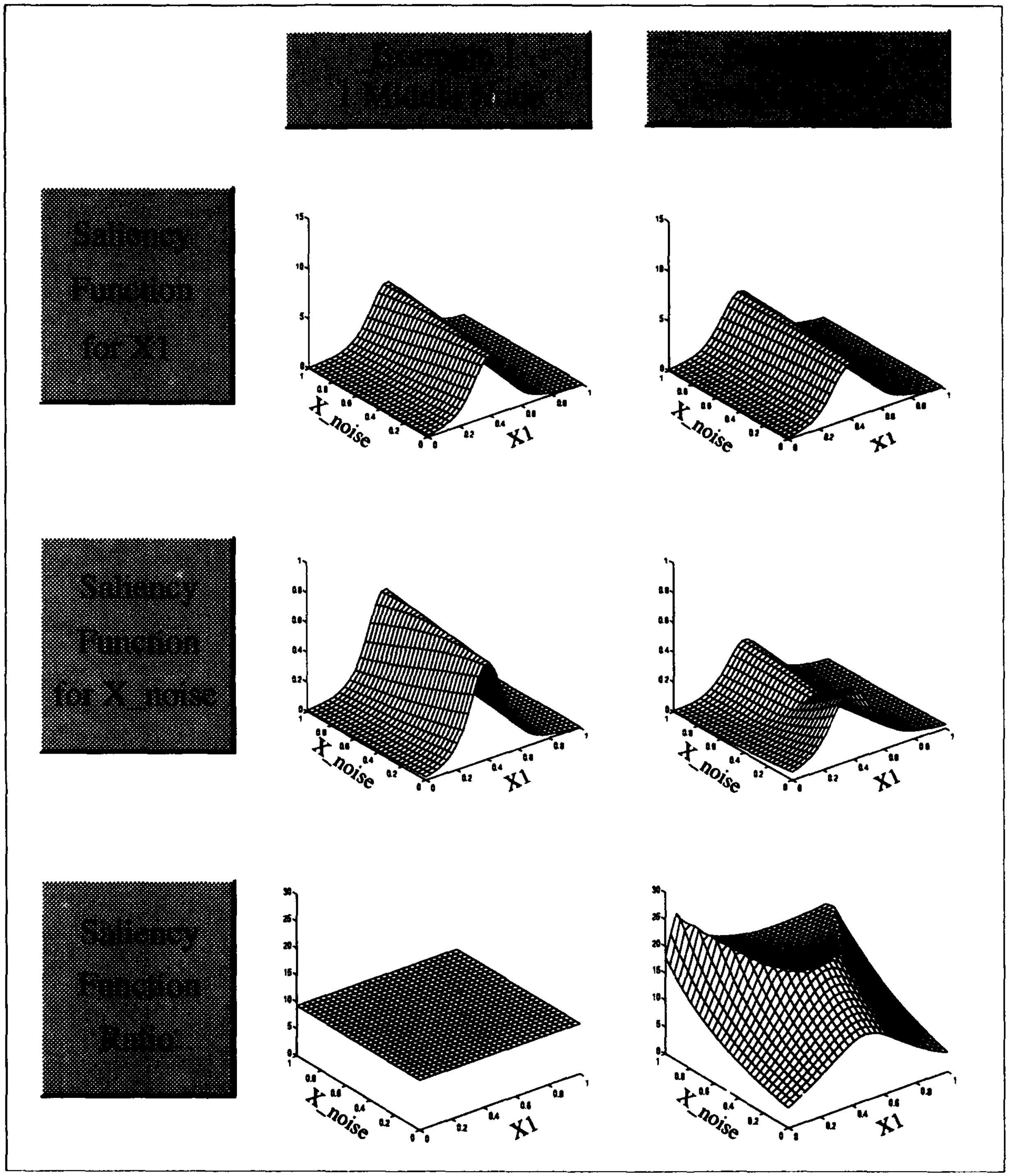


Figure 7. Example 1: 'Saliency Functions' for One and Eight Middle Nodes

Table 6. Summary of Derivative-Based Saliency Metrics

| Criteria | $\Lambda_i$ | $\hat{\Lambda}_i^{pseudo}$ | $\hat{\Lambda}_i^{random}$ | $\hat{\Lambda}_i^{data}$ |
|---|---|---|---|---|
| Computation | intractable | tractable | tractable | tractable |
| Sampling regions | includes regions of maximum sensitivity | may include regions of maximum sensitivity | may include regions of maximum sensitivity | • may include regions of maximum sensitivity<br>• includes only regions of likelihood |
| Number of Evaluations of $\sum_{k=1}^{K} \left\| \frac{\partial z_k}{\partial x_i} \right\|$ | $R^M$ | $PRM$ | $N^r$ | $P$ |
| Sensitivity to $R$ | not tested | not significant | not significant | N/A |
| Sensitivity to Redundant Middle Nodes | yes | yes | yes | yes |
| Sensitivity to Training | if there are redundant middle nodes | if there are redundant middle nodes | if there are redundant middle nodes | if there are redundant middle nodes |
| Tactical Decisions | $R$, middle nodes | $R$, middle nodes | $R$, middle nodes | middle nodes |
| Performance | good (when tractable) | good | good | good |

the third example for 1000 and 2000 epochs. In the second example, the 'saliency function' ratio corresponding to two middle nodes begins to decrease after 5000 epochs of training. However, 5000 epochs is not sufficient to affect the 'saliency function' ratio for more than two middle nodes. In the third example, a similar phenomena also occurs at 500 epochs of training.

The integrated saliency metric $\Lambda_i$ and the three approximations, $\hat{\Lambda}_i^{pseudo}$, $\hat{\Lambda}_i^{random}$, and $\hat{\Lambda}_i^{data}$ are summarized in Table 6.

*3.2.7 Summary.* In this section, a framework for derivative-based saliency is presented. Integrated saliency is introduced as a 'truth model' for derivative-based saliency. Three tractable approximations for integrated saliency are defined. The approximations were analyzed for sensitivities with respect to sampling, training, and redundant middle nodes.

All of the metrics are sensitive to the number of middle nodes and to the amount of training. It is important to minimize the number of redundant middle nodes, since the saliency metrics are most sensitive to the effects of training in the presence of redundant middle nodes. This is because the redundant middle nodes begin to incorporate unnecessary information from irrelevant features. After sufficient training, the extraneous parameter weights associated with redundant middle nodes will increase, which is indicative of data memorization. If the parameter weights do not increase proportionally with all the features, the saliency results become contaminated.

Although similar results are produced by all of these metrics, the known data metric $\hat{\Lambda}_i^{\mathrm{data}}$ in Equation 31 on page 68 is probably the best choice. This metric is measured in regions where the data is known, and it requires fewer calculations than the integrated and 'pseudo-sampling' metrics.

### *3.3 Relating Derivative and Weight-Based Saliency*

In this section, a mathematical connection is shown between the known derivative-based saliency, $\hat{\Lambda}_i^{\mathrm{data}}$, and weight-based saliency. First, a form of the weight-based saliency is defined. Then, the theoretical relationship between $\hat{\Lambda}_i^{\mathrm{data}}$ and the vector of weights emanating from feature $i$, $\hat{\mathbf{w}}_i^1$, is derived. This relationship is evaluated using 'saliency function' ratios for the three examples shown in Figure 5.

*3.3.1 Backgrˀund.* A weight-based saliency metric is suggested by Tarr [72:45]. Tarr conceived weight saliency based on the idea that weights connected to important features attain the largest values (absolute values); weights connected to less important features attain smaller values (absolute values); and weights connected to unimportant features would probably attain values somewhere near zero [72:45]. Tarr defined weight saliency as

$$\Upsilon_i = \sum_j (\hat{w}_{ij}^1)^2,$$

where $\hat{w}^1_{ij}$ denotes the $j$th element of $\hat{\mathbf{w}}^1_i$ or the estimated weight between the $i$th input feature and the $j$th hidden node. Tarr's definition of weight saliency is based on the Euclidean norm of the estimated weights associated with a feature input. A general definition of weight saliency based on the definition of the $r$th norm of a feature's estimated weights is given as

$$\Upsilon^r_i = \|\hat{\mathbf{w}}^1_i\|_r , \tag{32}$$

The effectiveness of weight-based saliency depends on two things [72:45].

1. $\mathbf{w}^1_i$ must be from a trained neural network of appropriate complexity.
2. The input features must be normalized to have approximately the same ranges.

Computationally, this metric is much simpler than other available saliency metrics. Tarr presents results which show $\Upsilon_i$ provides feature saliency rankings similar to $\hat{\Lambda}_i$ [72:49].

*3.3.2 Theoretical Relationship.* The derivative-based saliency, $\hat{\Lambda}^{\mathrm{data}}_i$, is defined as a function of the estimated weight parameters and the known training data. The estimated weight parameters are defined as a function of the known training data used to train the network. The derivative-based saliency, $\hat{\Lambda}^{\mathrm{data}}_i$, and the estimated weight parameters used to define weight saliency are interrelated. In this section, an upper bound is derived for $\hat{\Lambda}^{\mathrm{data}}_i$ which relates these quantities. The upper bound of $\hat{\Lambda}^{\mathrm{data}}_i$ for any feature $i$ is the vector product of a constant vector times a vector containing the absolute value of the estimated weight parameters associated with feature $i$ [70].

The term $\frac{\partial z_k}{\partial x_i}(\mathbf{x}, \hat{\mathbf{w}})$ of $\hat{\Lambda}^{\mathrm{data}}_i$ defined in Equation 31 on page 68 can be expanded as

$$\begin{aligned} \frac{\partial z_k}{\partial x_i}(\mathbf{x}, \hat{\mathbf{w}}) &= \frac{\partial}{\partial x_i}\left[ f_h(\sum_{j=0}^{H} x^1_j \hat{w}^2_{jk}) \right] \\ &= \delta^2_k \frac{\partial}{\partial x_i}\left[ \sum_{j=1}^{H} x^1_j \hat{w}^2_{jk} \right] \end{aligned}$$

$$
\begin{aligned}
&= \delta_k^2 \frac{\partial}{\partial x_i}\left[\sum_{j=1}^{H} f_k\left(\sum_{i=0}^{M} x_i \hat{w}_{ij}^1\right)\hat{w}_{jk}^2\right] \\
&= \delta_k^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jk}^2 \frac{\partial}{\partial x_i}\left[\sum_{i=0}^{M} x_i \hat{w}_{ij}^1\right] \\
&= \delta_k^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jk}^2 \hat{w}_{ij}^1,
\end{aligned}
$$

where definitions of $\delta_k^2$, $\delta_j^1$, $x_j^1$, $z_k$, $\hat{w}_{jk}^2$ and $\hat{w}_{ij}^1$ are reviewed in Section 2 of this chapter. Using this expression, $\hat{\Lambda}_i^{\text{data}}$ is defined as

$$
\hat{\Lambda}_i^{\text{data}} = P^{-1}\sum_{p=1}^{P}\sum_{k=1}^{K}\left|\delta_k^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jk}^2 \hat{w}_{ij}^1\right| \tag{33}
$$

The theoretical relationship between $\hat{\Lambda}_i^{\text{data}}$ and the estimated weight parameters associated with feature $i$ is developed by expanding $\hat{\Lambda}_i^{\text{data}}$ about the $K$ output nodes and then about the $H$ middle nodes.

$$
\begin{aligned}
\hat{\Lambda}_i^{\text{data}} &= P^{-1}\sum_{p=1}^{P}\sum_{k=1}^{K}\left|\delta_k^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jk}^2 \hat{w}_{ij}^1\right| \\
&= P^{-1}\sum_{p=1}^{P}[\left|\delta_1^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{j1}^2 \hat{w}_{ij}^1\right| + \cdots + \left|\delta_K^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jK}^2 \hat{w}_{ij}^1\right|] \\
&= P^{-1}\sum_{p=1}^{P}[|\delta_1^2\delta_1^1\hat{w}_{11}^2\hat{w}_{i1}^1 + \cdots + \delta_1^2\delta_H^1\hat{w}_{H1}^2\hat{w}_{iH}^1| + \cdots + \\
&\quad |\delta_K^2\delta_1^1\hat{w}_{1K}^2\hat{w}_{i1}^1 + \cdots + \delta_K^2\delta_H^1\hat{w}_{HK}^2\hat{w}_{iH}^1|] \\
&\leq P^{-1}\sum_{p=1}^{P}[|\delta_1^2\delta_1^1\hat{w}_{11}^2\hat{w}_{i1}^1| + \cdots + |\delta_1^2\delta_H^1\hat{w}_{H1}^2\hat{w}_{iH}^1| + \cdots + \\
&\quad |\delta_K^2\delta_1^1\hat{w}_{1K}^2\hat{w}_{i1}^1| + \cdots + |\delta_K^2\delta_H^1\hat{w}_{HK}^2\hat{w}_{iH}^1|] && (34) \\
&= P^{-1}\sum_{p=1}^{P}[|\delta_1^2\delta_1^1\hat{w}_{11}^2|\,|\hat{w}_{i1}^1| + \cdots + |\delta_1^2\delta_H^1\hat{w}_{H1}^2|\,|\hat{w}_{iH}^1| + \cdots + \\
&\quad |\delta_K^2\delta_1^1\hat{w}_{1K}^2|\,|\hat{w}_{i1}^1| + \cdots + |\delta_K^2\delta_H^1\hat{w}_{HK}^2|\,|\hat{w}_{iH}^1|] && (35) \\
&= P^{-1}\sum_{p=1}^{P}[\{|\delta_1^2\delta_1^1\hat{w}_{11}^2| + \cdots + |\delta_K^2\delta_1^1\hat{w}_{1K}^2|\}\,|\hat{w}_{i1}^1| + \cdots + \\
&\quad \{|\delta_1^2\delta_H^1\hat{w}_{H1}^2| + \cdots + |\delta_K^2\delta_H^1\hat{w}_{HK}^2|\}\,|\hat{w}_{iH}^1|] && (36)
\end{aligned}
$$

An inequality sign replaced the equality sign for Equation 34 when the triangle inequality [38:92] was used to decouple the outputs. For a given input exemplar $p$ and middle node $j$, the result in the brackets $\{\cdot\}$, prior to each $\left|\hat{w}^1_{ij}\right|$ in Equation 36 is the same regardless of which feature is being examined. Therefore, a constant $\phi^p_j$ is substituted into Equation 36 for the quantity in the brackets $\{\cdot\}$ prior to each $\left|\hat{w}^1_{ij}\right|$ giving

$$\hat{\Lambda}^{\text{data}}_i \leq P^{-1} \sum_{p=1}^{P} [\phi^p_1 \left|\hat{w}^1_{i1}\right| + \cdots + \phi^p_H \left|\hat{w}^1_{iH}\right|] \tag{37}$$

Now, since $|\hat{w}^1_{iH}|$ is independent of $p$, then replacing $\sum_{p=1}^{P} \phi^p_j$ with a new constant $\Phi_j$ and rearranging terms gives

$$\hat{\Lambda}^{\text{data}}_i \leq P^{-1} \left|\hat{w}^1_{i1}\right| \Phi_1 + \cdots + P^{-1} \left|\hat{w}^1_{iH}\right| \Phi_H \tag{38}$$

Now, let $|\hat{\mathbf{w}}^1_i|$ be an $H$-dimensional vector containing the absolute values of the weights associated with the $i$th feature, (i.e. $|\hat{\mathbf{w}}^1_i| = [|\hat{w}^1_{i1}|, \cdots, |\hat{w}^1_{iH}|]'$), and let $\mathbf{\Phi}$ be an $H$-dimensional vector of constants associated with the middle nodes (i.e. $\mathbf{\Phi} = P^{-1}[\Phi_1, \cdots, \Phi_H]'$). The vector of constants $\mathbf{\Phi}$ is independent of $i$. Therefore, the known derivative-based saliency for the $i$th feature is bounded above by a constant linear combination of the vector $|\hat{\mathbf{w}}^1_i|$. That is

$$\hat{\Lambda}^{\text{data}}_i \leq \mathbf{\Phi}' \left|\hat{\mathbf{w}}^1_i\right| \tag{39}$$

*3.3.3 Analysis.* To study this mathematical connection, the weight saliency metric $\Upsilon^r_i$, for $r$ equal to one, two, and infinity, is compared to the saliency metric $\hat{\Lambda}^{\text{data}}_i$ for sensitivity to training and redundant middle nodes. Figure 8 shows average 'saliency function' ratio results over 30 neural networks for the three two-class multivariate examples summarized in Figure 5. The networks were trained for 100, 80, and 100 epochs, respectively. The $x$-axis represents the number of middle nodes and the $y$-axis corresponds to the 'saliency function' ratio of $x_1$ to $x_{\text{noise}}$. For each experiment, the average 'saliency function' ratio for known data is plotted against the average 'saliency function'

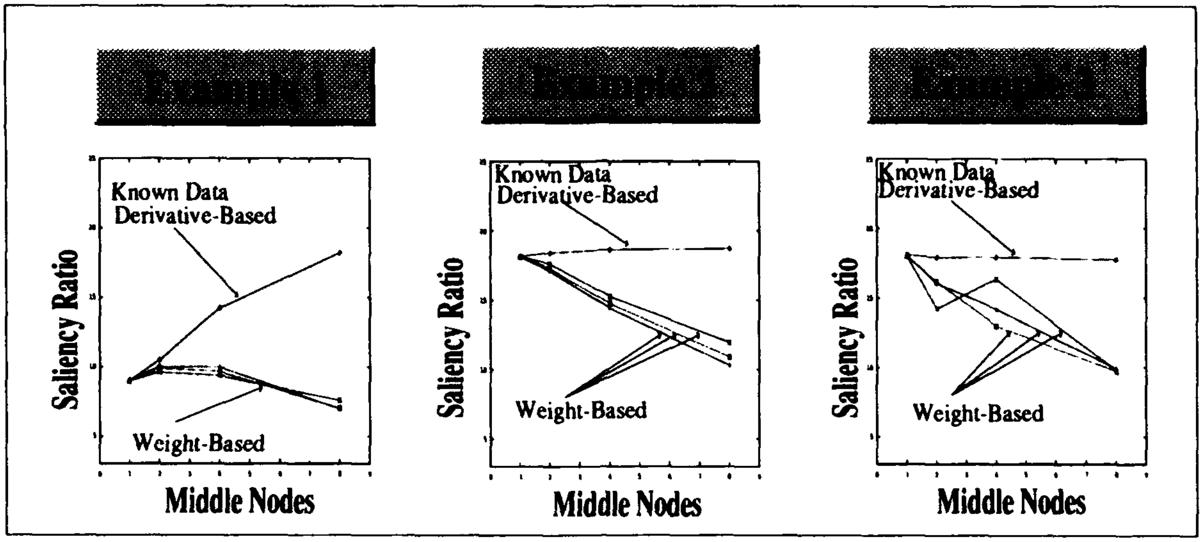


Figure 8. Summary of Derivative versus Weight Saliency

ratios for weight saliency. In Figure 8, it can be seen that the weight-based metric produces approximately the same ratio as the known data saliencies when one middle node is used. This occurs for two reasons:

1. The metric $\hat{\Lambda}_i^{\text{data}}$ is bounded above by $\mathbf{\Phi}' \, |\hat{\mathbf{w}}_i^1|$, where $|\hat{\mathbf{w}}_i^1|$ is a $H$-dimensional vector of the absolute value of the weights from feature $i$ to the $H$ middle nodes, and $\mathbf{\Phi}$ is a $H$-dimensional vector of constants associated with the $H$ middle nodes.

2. Only one middle node is used to train the neural network. Therefore, the constant term $\mathbf{\Phi}$ in $\mathbf{\Phi}' \, |\hat{\mathbf{w}}_i^1|$ cancels when a ratio is taken. When there is just one middle node, the ratio of $\hat{\Lambda}_1^{\text{data}}$ to $\hat{\Lambda}_{\text{noise}}^{\text{data}}$ is equal to the ratio of $|\hat{\mathbf{w}}_1^1|$ to $|\hat{\mathbf{w}}_{\text{noise}}^1|$, since the triangle inequality in Equation 34 is not needed.

For more than one middle node, the ratio of the weight saliencies is always smaller than the ratio of the derivative-based saliency. With additional middle nodes the network is over-parameterized. As middle nodes are added, the weights associated with $x_{\text{noise}}$ increase faster than the weights associated with $x_1$. Due to the over-parameterization, the parameters between $x_{\text{noise}}$ and the redundant middle nodes incorporate unnecessary information about the feature $x_{\text{noise}}$. This behavior can be associated with over-training. The sensitivity of the 'saliency function' ratios to increased middle nodes is revisited in Section 3.5 for a 'real world' problem. A final observation is that the similarity in the saliency rankings and 'saliency function' ratios of the collection of weight-based saliencies indicates that the choice of $r$ for $\Upsilon_i^r$ makes no appreciable difference.

*3.3.4 Summary.* The theoretical relationship between $\hat{\Lambda}_i^{\text{data}}$ and $|\hat{\mathbf{w}}_i^1|$ provides a mathematical connection between the metrics $\hat{\Lambda}_i^{\text{data}}$ and $\Upsilon_i^r$:

- $\hat{\Lambda}_i^{\text{data}}$ is a vector product of a constant vector and the vector $|\hat{\mathbf{w}}_i^1|$
(i.e. it is a linear combination of $|\hat{\mathbf{w}}_i^1|$)
- $\Upsilon_i^r$ corresponds to the $r$th norm of $\hat{\mathbf{w}}_i^1$ (or $|\hat{\mathbf{w}}_i^1|$)

An analysis of this relationship shows that the relative weight-based saliencies are equal to the relative derivative-based saliencies for neural networks with one middle node. In the presence of additional redundant middle nodes, empirical results encompass a range of two-class multivariate examples (from class distributions not overlapping to class distributions significantly overlapping) indicates that the relative weight-based saliencies are smaller than the relative derivative-based saliencies.

### *3.4 $P_e$-based Feature Saliency Metrics*

In this section, neural network $P_e$-based feature saliency metrics are discussed. These metrics are appropriate for classification problems, but not for regression problems. Probability of error metrics are developed for feedforward neural networks which approximate a Bayesian optimal discriminant. The assumptions necessary for this approximation are discussed in Chapter I. There are several contributions in the area of $P_e$ metrics presented in this section.

In this section, the background on $P_e$-based feature saliency metrics is reviewed. Then, an exact relationship is shown between a $P_e$-based metric and $\hat{\Lambda}_i$. Next, a new $P_e$-based neural network feature metric $\Gamma_i$ is defined using a restricted subset of the terms associated with $\hat{\Lambda}_i^{\text{data}}$. Two results related to $\Gamma_i$ are derived. One, the relationship between $\Gamma_i$ and $\hat{\Lambda}_i^{\text{data}}$ is derived, and two, an upper bound for $\Gamma_i$ is derived. Analysis is presented which compares the metrics $\Gamma_i$ and $\hat{\Lambda}_i^{\text{data}}$, and $\hat{\Lambda}_i^{\text{pseudo}}$ for both a two class and a four class problem. Finally, the results presented in this section are summarized.

*3.4.1 Background.* The $P_e$ feature evaluation metric is commonly used whenever the goal is minimizing classifier error rate with features of equal measurement cost. As a result, probability of error is often used as a bench mark for independently measuring the classification error associated with using either a single feature or a set of features for classification neural networks [51, 59, 60].

The $P_e$ metric defined in Section 2.4.1 page 38 Equation 16 is defined again here for convenience,

$$P_e(\mathbf{x}) = E_{\mathbf{x}}[1 - \max\{P(C_1|\mathbf{x}), \cdots, P(C_K|\mathbf{x})\}] \quad , \tag{40}$$

where $\mathbf{x}$ is the vector of features for which $P_e$ is measured, $E_{\mathbf{x}}[\,\cdot\,]$ is the expectation operator, and $P(C_k|\mathbf{x})$, is the posterior probability of class $k$ for $\mathbf{x}$, defined as

$$P(C_k|\mathbf{x}) = \frac{P(C_k)P_k(\mathbf{x})}{\sum_{k=1}^{K} P(C_k)P_k(\mathbf{x})} \tag{41}$$

where $P(C_k)$ is the prior probability of class $k$ and $P(\mathbf{x}|C_k)$ is the class conditional probability function of $\mathbf{x}$ for class $k$.

By definition, $\sum_{k=1}^{K} P(C_k|\mathbf{x}) = 1$. Also, the class specific probability of error associated with the $k$th class is given as $P_e(C_k, \mathbf{x})$, is given as:

$$
\begin{aligned}
P_e(C_k, \mathbf{x}) &= 1 - P(C_k|\mathbf{x}) && (42)\\
&= \sum_{l \neq k}^{K} P(C_l|\mathbf{x}) && (43)
\end{aligned}
$$

Under certain necessary conditions, the feedforward neural network approximates a Bayes optimal discriminant function in the limit (see Section 1.2 of Chapter I). The implications for interpreting a trained feedforward neural network in the limit as an approximation for a Bayes optimal discriminant function are that classical definitions associated with probability of error given in Equations 40, 41, and 42 can be redefined as an approximation in neural network terms. Specifically, using the fact that $z_k(\mathbf{x}, \hat{\mathbf{w}}) \approx P(C_k|\mathbf{x})$, the neural network approximations to the class specific probability of error $P_e(C_k, \mathbf{x})$ and classifier probability of error $P_e(\mathbf{x})$ are defined in Equations 45 and 49, respectively.

Priddy illustrates a relationship between class specific probability of error $P_e(k, \mathbf{x})$ and the derivative-based feature saliency metric, $\hat{\Lambda}_i$ defined in Equation 29 [51]. This relationship relies on the assumptions necessary for feedforward neural networks to approximate a Bayes optimal discriminant function in the limit. Priddy's defines a Bayesian-based saliency metric, $\Omega_i$, as

$$\Omega_i = \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} \left| \frac{\partial \hat{P}_e(C_k, \mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})}{\partial x_i} \right|, \qquad (44)$$

where $P$ is the number of training vectors $\mathbf{x}$; $M$ is the number of features; $R$ is the number of uniformly spaced points covering the range of each input feature found in the training set; $K$ is the number of output classes; the vector $\mathbf{x}^p_{m(r)}$ is the $p$th exemplar $\mathbf{x}^p$ with its $m$th component replaced by, $d_r$ the $r$th component of $\mathbf{d}_m$ defined in Equation 28; $(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})$ indicates that the derivative is evaluated with the feature vector $\mathbf{x}^p_{m(r)}$ and the final estimates of the trained network weight parameters $\mathbf{w}$; and $\hat{P}_e(k, \mathbf{x}, \hat{\mathbf{w}})$ is a neural network approximation to the class specific probability of error.

Priddy defines the neural network approximation for class specific probability of error $\hat{P}_e(C_k, \mathbf{x}, \hat{\mathbf{w}})$ as

$$\hat{P}_e(C_k, \mathbf{x}, \hat{\mathbf{w}}) = \sum_{l \neq k}^{K} z_l(\mathbf{x}, \mathbf{w}) \tag{45}$$

which is similar to Equation 43. Now, substituting Equation 45 into Equation 44 results in

$$\Omega_i = \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} \left| \sum_{l \neq k}^{K} \frac{\partial z_l(\mathbf{x}, \hat{\mathbf{w}})}{\partial x_i} \right| \tag{46}$$

Using the triangle inequality, Priddy proves that $\Omega_i$ is bounded above by a simplified saliency metric $\hat{\Omega}_i$ [51], i.e.

$$\Omega_i \leq \hat{\Omega}_i, \tag{47}$$

where

$$\hat{\Omega}_i = \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} \sum_{l \neq k}^{K} \left| \frac{\partial z_j(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})}{\partial x_i} \right|$$

The Bayesian-based metric $\hat{\Omega}_i$ is related to the metric $\hat{\Lambda}_i$, since it involves the partials of $z_k$ with respect to $x_i$ and the pseudo-sampling of unknown vectors from the feature space [59]. Priddy shows $\hat{\Omega}_i$ is a scalar multiple of the $\hat{\Lambda}_i$ in Equation 29:

$$\hat{\Omega}_i = (K - 1)\hat{\Lambda}_i, \tag{48}$$

where $K$ is the total number of output classes [51]. Therefore, the two saliency metrics, $\hat{\Omega}_i$ and $\hat{\Lambda}_i$, produce identical feature rankings.

*3.4.2 Equality of Two $P_e$-based Metrics.* In this section, it is shown that the metric $\Omega_i$ is exactly equal to the metric $\hat{\Lambda}_i$. Using the relationship shown in Equation 42, a neural network approximation to class specific probability of error can also be defined as

$$\hat{P}_e(C_k, \mathbf{x}, \hat{\mathbf{w}}) = 1 - z_k(\mathbf{x}, \hat{\mathbf{w}}). \tag{49}$$

Now substituting Equation 49 into Equation 44 and simplifying illustrates the equality of $\Omega_i$ and $\hat{\Lambda}_i$:

$$\begin{aligned}
\Omega_i &= \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{r=1}^{R}\sum_{k=1}^{K}\left|\frac{\partial \hat{P}_e(k, \mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})}{\partial x_i}\right| \\
&= \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{r=1}^{R}\sum_{k=1}^{K}\left|\frac{\partial \left[1 - z_k(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})\right]}{\partial x_i}\right| \\
&= \sum_{p=1}^{P}\sum_{m=1}^{M}\sum_{r=1}^{R}\sum_{k=1}^{K}\left|\frac{\partial z_k(\mathbf{x}^p_{m(r)}, \hat{\mathbf{w}})}{\partial x_i}\right| \\
&= \hat{\Lambda}_i
\end{aligned} \tag{50}$$

The relationship between $\Omega_i$ and $\hat{\Lambda}_i$ is derived exactly without recourse to $\hat{\Omega}_i$ of Equation 47 [51].

*3.4.3 Derivation of a New $P_e$-based Metric.* The definition of $P_e$ reviewed in Equation 40 is used to derive a new Bayesian-based saliency metric. This metric is related closely to the neural network approximation to the Bayesian classification error $P_e(\mathbf{x})$ which is defined in neural network terms as

$$\begin{aligned}
\hat{P}_e(\mathbf{x}, \hat{\mathbf{w}}) &= P^{-1}\sum_{p=1}^{P}[1 - \max\{z_1(\mathbf{x}^p, \hat{\mathbf{w}}), \cdots, z_K(\mathbf{x}^p, \hat{\mathbf{w}})\}] && (51) \\
&= P^{-1}\sum_{p=1}^{P}\left[\hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})\right] && (52)
\end{aligned}$$

where $\hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})$ is the probability of error associated with the $p$th exemplar from a set of $P$ total exemplars. The new Bayesian-based saliency metric for the $i$th feature is defined using $\hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})$

$$\Gamma_i = P^{-1}\sum_{p=1}^{P}\left|\frac{\partial \hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right|, \tag{53}$$

Like $\hat{\Lambda}_i^{\text{data}}$, this metric depends only on the known data. Let $z_{k_{\max}}(\mathbf{x}^p, \hat{\mathbf{w}})$ be a function which is given as

$$z_{k_{\max}}(\mathbf{x}^p, \hat{\mathbf{w}}) = \max\{z_1(\mathbf{x}^p, \hat{\mathbf{w}}), \cdots, z_K(\mathbf{x}^p, \hat{\mathbf{w}})\}$$

where $k_{\max}$ represents the subscript $k$ associated with the $\max\{z_1(\mathbf{x}^p, \hat{\mathbf{w}}), \cdots, z_K(\mathbf{x}^p, \hat{\mathbf{w}})\}$. Using $z_{k_{\max}}$, $\hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})$ in Equation 53 becomes

$$\hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}}) = 1 - z_{k_{\max}}(\mathbf{x}^p, \hat{\mathbf{w}}) \tag{54}$$

giving:

$$\Gamma_i = P^{-1} \sum_{p=1}^{P} \left| \frac{\partial\ [z_{k_{\max}}(\mathbf{x}^p, \hat{\mathbf{w}})]}{\partial\ x_i} \right|$$

*3.4.4 Theoretical relationships.* The relationship between $\Gamma_i$ and the derivative-based saliency $\hat{\Lambda}_i^{\text{data}}$ is derived from the definition of $\hat{\Lambda}_i^{\text{data}}$:

$$\begin{aligned}
\hat{\Lambda}_i^{\text{data}} &= P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} \left| \frac{\partial\, z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial\, x_i} \right| \\
&= P^{-1} \sum_{p=1}^{P} \left| \frac{\partial\, z_{k_{\max}}(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial\, x_i} \right| + P^{-1} \sum_{p=1}^{P} \sum_{k \neq k_{\max}}^{K} \left| \frac{\partial\, z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial\, x_i} \right| \\
&= P^{-1} \sum_{p=1}^{P} \left| \frac{\partial\, [1 - \max\{z_1(\mathbf{x}^p, \hat{\mathbf{w}}), \cdots, z_K(\mathbf{x}^p, \hat{\mathbf{w}})\}]}{\partial\, x_i} \right| + P^{-1} \sum_{p=1}^{P} \sum_{k \neq k_{\max}}^{K} \left| \frac{\partial\, z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial\, x_i} \right| \\
&= P^{-1} \sum_{p=1}^{P} \left| \frac{\partial\, \hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial\, x_i} \right| + P^{-1} \sum_{p=1}^{P} \sum_{k \neq k_{\max}}^{K} \left| \frac{\partial\, z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial\, x_i} \right| \\
&= \Gamma_i + P^{-1} \sum_{p=1}^{P} \sum_{k \neq k_{\max}}^{K} \left| \frac{\partial\, z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial\, x_i} \right|
\end{aligned}$$

In summary, the exact relationship between $\Gamma_i$ and $\hat{\Lambda}_i^{\text{data}}$ is

$$\Gamma_i = \hat{\Lambda}_i^{\text{data}} - P^{-1} \sum_{p=1}^{P} \sum_{k \neq k_{\max}}^{K} \left| \frac{\partial\, z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial\, x_i} \right| \tag{55}$$

It can also be shown that $\frac{1}{2}\hat{\Lambda}_i^{\text{data}}$ is an upper bound for $\Gamma_i$. This relationship is derived using the triangle inequality on the second term of Equation 55 in concert with the Bayesian relationship that $\sum_{k=1}^{K} z_k(\mathbf{x}, \hat{\mathbf{w}}) = 1$ as follows:

$$\begin{aligned}
\Gamma_i &= \hat{\Lambda}_i^{\text{data}} - P^{-1}\sum_{p=1}^{P} \sum_{k \neq k_{\max}}^{K} \left|\frac{\partial z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right| \\
\Gamma_i &\leq \hat{\Lambda}_i^{\text{data}} - P^{-1}\sum_{p=1}^{P} \left|\sum_{k \neq k_{\max}}^{K} \frac{\partial z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right| \\
\Gamma_i &\leq \hat{\Lambda}_i^{\text{data}} - P^{-1}\sum_{p=1}^{P} \left|\frac{\partial \sum_{k \neq k_{\max}}^{K} z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right| \\
\Gamma_i &\leq \hat{\Lambda}_i^{\text{data}} - P^{-1}\sum_{p=1}^{P} \left|\frac{\partial z_{k_{\max}}(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right| \\
\Gamma_i &\leq \hat{\Lambda}_i^{\text{data}} - P^{-1}\sum_{p=1}^{P} \left|\frac{\partial [1 - \max\{z_1(\mathbf{x}^p, \hat{\mathbf{w}}), \cdots, z_K(\mathbf{x}^p, \hat{\mathbf{w}})\}]}{\partial x_i}\right| \\
\Gamma_i &\leq \hat{\Lambda}_i^{\text{data}} - P^{-1}\sum_{p=1}^{P} \left|\frac{\partial \hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right| \\
\Gamma_i &\leq \hat{\Lambda}_i^{\text{data}} - \Gamma_i \\
\Gamma_i &\leq \frac{1}{2}\,\hat{\Lambda}_i^{\text{data}}
\end{aligned} \tag{56}$$

For a two class problem, the new metric is at its upper bound exactly. That is: $\Gamma_i = \frac{1}{2}\,\hat{\Lambda}_i^{\text{data}}$, since

$$P^{-1}\sum_{p=1}^{P} \sum_{k \neq k_{\max}}^{K} \left|\frac{\partial z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right| = P^{-1}\sum_{p=1}^{P} \left|\sum_{k \neq k_{\max}}^{K} \frac{\partial z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right|$$

when $K = 2$. This means that for a two class problem

$$\left|\frac{\partial \hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right| = \frac{1}{2}\left|\frac{\partial \hat{P}_e(k, \mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}\right|$$

For more than a two class problem, the metric $\Gamma_i$ will be at its upper bound only if the partial derivatives $\frac{\partial z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i}$ for $k \neq k_{\max}$ are all the same sign. To analyze the partial derivatives, Equation 25 is shown again below for convenience as

$$\frac{\partial z_k(\mathbf{x}^p, \hat{\mathbf{w}})}{\partial x_i} = \delta_k^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jk}^2 \hat{w}_{ij}^1$$

For all $k$, the following is true:

- $\delta_k^2 \geq 0$
- $\delta_j^1 \geq 0$
- $\hat{w}_{ij}^1$ are constants

Therefore, it is the middle node to output weights $w_{jk}^2$ which influence whether the partial derivatives will be the same sign for all outputs. For a net with just one middle node, all the derivatives will be the same sign if the weights $w_{1k}^2$ are the same sign for all $k \neq k_{\max}$.

*3.4.5 Analysis.* The upper bound for $\Gamma_i$ is investigated empirically with a two class and a four class problem. For the two class problem, the XOR problem is used. The exclusive-or (XOR) problem illustrated in Figure 9 is a standard benchmark problem used with neural networks. In this nonlinear classification problem, no single line can be drawn to separate class 1 and class 2 regions. Five hundred data points are randomly generated for the XOR problem. A training set of 400 and a training-test set of 100 are used.

Saliency metric results for $\hat{\Lambda}_i^{\mathrm{data}}$ and $\Gamma_i$ are summarized for 30 'trained' neural networks which were trained with the same data set, but with different random initial weights and a different random order of training vector presentation. The neural networks used four middle nodes. Log-linear declining learning rates were used to improve the neural network's convergence to a solution. For all runs, 700 epochs were used. Also, a seven percent minimum training set classification error was required for the network's solution to be considered from a 'trained' network. In total, 51 networks were trained; 21 networks did not meet the seven percent requirement. The average

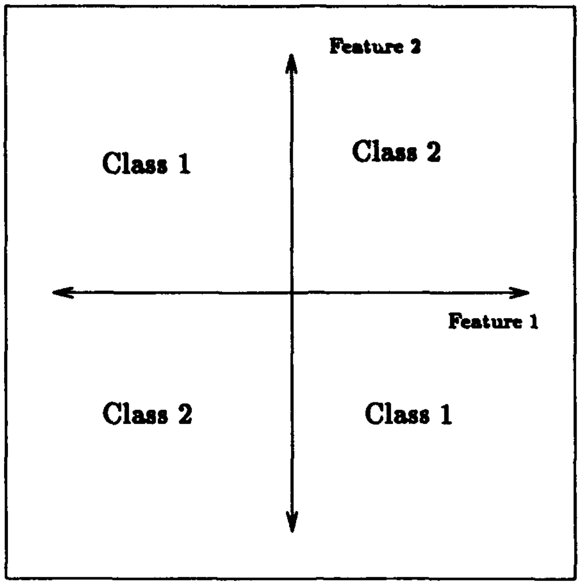


Figure 9. The XOR Problem

reprinted from [5]

Table 7. XOR Problem: Saliency Metric Means for 30 Trained Networks

| Feature | $\frac{1}{2}\hat{\Lambda}_i^{\text{data}}$ | $\Gamma_i$ |
|---|---|---|
| $x$ | 1.001 | 1.001 |
| $y$ | 1.078 | 1.078 |
| bias | 0.84 | 0.84 |
| training data used | | |

training and test set classification errors for the remaining 30 neural networks were 1.59 and 2.96 percent, respectively. The saliency was computed for the 30 networks at 700 epochs. As expected, the metric $\Gamma_i$ (to within roundoff error) is exactly equal to its upper bound of $\frac{1}{2}\hat{\Lambda}_i^{\text{data}}$. This is demonstrated with the XOR problem in Table 7.

Table 8. Four Class Problem: Saliency Metric Means for 30 Trained Networks

| Feature | $\frac{1}{2}\hat{\Lambda}_i^{\mathrm{data}}$ | $\Gamma_i$ |
|---|---|---|
| $x$ | 0.879 | 0.843 |
| $y$ | 0.567 | 0.606 |
| bias | 0.360 | 0.387 |
| training data used | | |

The metric $\Gamma_i$ is, generally, less than its upper bound when used with more than a two class problem. A four class problem with two variables is studied. The classes are multivariate normally distributed with an identity matrix for the covariance matrix. The mean vectors for the four classes are: (4.5, 2.17), (2.0, 6.5), (7.0, 6.5), and (12.0, 6.5). Five hundred data vectors are randomly generated for this problem: 400 for the training set and 100 for the training-test set. Again, saliency metric results are summarized for 30 'trained' neural networks trained with the same data set. For all runs, a minimal network of two middle nodes (determined from a number of pilot simulations), a log-linear declining learning rate, and 500 epochs were used. Also, a five percent minimum training set classification error was required for the network's solution to be considered from a 'trained' network. For this problem, 30 networks were trained, and all the networks met the five percent requirement. The average training and test set classification errors for the remaining 30 neural networks were 2.06 and 5.77 percent, respectively. The saliency was computed for the 30 networks at 500 epochs. The metric $\hat{\Lambda}_i^{\mathrm{data}}$ and $\Gamma_i$ are shown in Table 8.

*3.4.6 Summary.* Neural network Bayesian-based feature saliency metrics are covered in this section. These metrics are developed for use with neural networks which approximate a Bayesian optimal discriminant in the limit, and they are only appropriate for use with classification problems.

The research contributions in this section are:

- An exact relationship is shown between Ruck's metric $\hat{\Lambda}_i$ and the Bayesian-based metric suggested by Priddy.
- A new Bayesian-based neural network feature metric $\Gamma_i$ is defined using only a subset of the terms in $\hat{\Lambda}_i^{\text{data}}$.
- The relationship between $\Gamma_i$ and $\hat{\Lambda}_i^{\text{data}}$ is derived.
- An upper bound for $\Gamma_i$ is derived.

For classification applications, the metric $\Gamma_i$ is more appealing than $\hat{\Lambda}_i^{\text{data}}$. The metric $\Gamma_i$ is developed from classifier error $\hat{P}_e(\mathbf{x}, \hat{\mathbf{w}})$ (see Equation 51 on page 86), rather than class specific error $\hat{P}_e(k, \mathbf{x}, \hat{\mathbf{w}})$ (see Equation 45 on page 85). Also, the saliency metric $\Gamma_i$ is computed using only a subset of the terms used for $\hat{\Lambda}_i^{\text{data}}$ making the definition of $\Gamma_i$ more succinct than the definition of $\hat{\Lambda}_i^{\text{data}}$.

### *3.5 Unifying Theoretical Relationships*

*3.5.1 Introduction.* This section documents the relationships between the set of available neural network feature saliency metrics. Neural network feature saliency metrics include the established metrics introduced in Chapter II and three new metrics defined in this chapter: $\Lambda_i$, $\hat{\Lambda}_i^{\text{data}}$ and $\Gamma_i$.

The derivative and weight-based feature saliency metrics are defined in Table 9. Formal definitions and applicable references are presented for each metric. With the exception of the weight-based saliencies, each of the feature saliency metrics is defined in Table 9 as a function of $g_i$. The metrics differ in the derivative that is taken and in the definition of $g_i$ which is used.

The function $g_i$, also given in Table 9 can be interpreted as the absolute value of some form of a derivative of a network error function. For the metrics $\Lambda_i$, $\hat{\Lambda}_i$, $\hat{\Lambda}_i^{\text{data}}$, and $\hat{\Omega}_i$, the network error function is defined $d_k - z_k(\mathbf{x}, \hat{\mathbf{w}})$, and the derivative is taken with respect to the feature of interest $x_i$. For the metric $\rho_i$, the network error function is the same, but the derivative is taken with respect to a relevance function $\alpha_i$ for the feature of interest. For the metric $\Omega_i$, the network error function is the approximate probability of error for the $k$th network output for the $p$th input exemplar, i.e. $\hat{P}_e(k, \mathbf{x}^p, \hat{\mathbf{w}})$, and the derivative is taken with respect to the feature of interest $x_i$. For the metric $\Gamma_i$, the network error function is the approximate probability of error for the network given the $p$th input exemplar , i.e. $\hat{P}_e(\mathbf{x}^p, \hat{\mathbf{w}})$, and the derivative is taken with respect to the feature of interest $x_i$. For the metric $\bar{s}_i$, the error function is the squared error defined $[d_k - z_k(\mathbf{x}, \hat{\mathbf{w}})]^2$, and here a form of the Taylor Series approximation to the total derivative is used. In Table 10, detailed notational saliency definitions, as well as established theoretical relationships among the saliency metrics are presented.

Since previous examples have been contrived problems, a 'real world' problem is analyzed in this section. A description of this problem followed by a comparison and evaluation of the various feature saliency metrics follows.

*3.5.2 Background.* The 'real world' problem is a two class problem using forward looking radar (FLIR) data to discriminate targets from non-targets. The targets consisted of tanks, trucks, and armored personnel carriers. Nine features were used based on previous application experience by Roggemann [56, 57, 58] and by Ruck [60]. A description for the nine FLIR features is given in Table 11.

*3.5.3 Analysis.* Analysis of the FLIR problem is discussed in this section. Saliency metric ranks, means, and standard deviations are documented for all of the metrics presented in Table 9, except the integrated metric $\Lambda_i$. Results are not computed for $\Lambda_i$, because it is not computationally tractable for this problem. The average network accuracy is documented as the features are

## Table 9. Saliency Metric Definitions

| Saliency Metric | Reference | Metric Definition | Function $g_i$ Definition |
|---|---|---|---|
| $\Lambda_i$ | Section 3.2 | $\frac{\int_{\mathcal{R}} \int_{f_i=0}^{f_i=\sum_{k=1}^{K} g_i} f_i df_i d\mathcal{R}}{V_i}$ ,<br>where $V_i$ is defined in Equation 27<br>$\mathcal{R}$ represents the feature space region | $g_i = \left\lvert \frac{\partial z_k(\mathbf{x},\mathbf{w})}{\partial w_i} \right\rvert$ |
| $\hat{\Lambda}_i$ | [59] | $\sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} g_i$ | $g_i = \left\lvert \frac{\partial z_k(\mathbf{x}^p_{m(r)},\mathbf{w})}{\partial w_i} \right\rvert$ |
| $\hat{\Lambda}_i^{pseudo}$ | Section 3.2 | $(PMR)^{-1} \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} g_i$ | $g_i = \left\lvert \frac{\partial z_k(\mathbf{x}^p_{m(r)},\mathbf{w})}{\partial w_i} \right\rvert$ |
| $\hat{\Lambda}_i^{data}$ | Section 3.2 | $P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} g_i$ | $g_i = \left\lvert \frac{\partial z_k(\mathbf{x},\mathbf{w})}{\partial w_i} \right\rvert$ |
| $\Omega_i$ | [51] and Section 3.4 | $\sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} g_i$ | $g_i = \left\lvert \frac{\partial \hat{P}_c(k,\mathbf{x}^p_{m(r)},\mathbf{w})}{\partial w_i} \right\rvert$ ,<br>where $\hat{P}_c(k,\mathbf{x}^p_{m(r)},\mathbf{w})$ is defined in Equation 45 |
| $\hat{\Omega}_i$ | [51] | $\sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{k=1}^{K} \sum_{l \neq k}^{K} g_i$ | $g_i = \left\lvert \frac{\partial z_k(\mathbf{x}^p_{m(r)},\mathbf{w})}{\partial w_i} \right\rvert$ |
| $\Gamma_i$ | Section 3.4 | $P^{-1} \sum_{p=1}^{P} g_i$ | $g_i = \left\lvert \frac{\partial \hat{P}_c(\mathbf{x}^p,\mathbf{w})}{\partial w_i} \right\rvert$ ,<br>where $\hat{P}_c(\mathbf{x}^p,\mathbf{w})$ is defined with Equation 54 |
| $\rho_i$ | Skeletonization [45]<br>Appendix A | $P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} g_i$ | $g_i = \left\lvert \frac{\partial z_k(\mathbf{x}^p,\mathbf{w})}{\partial \alpha_i} \right\rvert$ ,<br>where $\alpha_i$ is a relevance factor associated with feature $i$ |
| $s_i$ | Optimal brain damage [39]<br>Appendix A | $P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} g_i$ | $g_i = d\mathcal{E}$, where $\mathcal{E} = [d_k - z_k(\mathbf{x}^p,\mathbf{w})]^2$<br>Concerned with change in $\mathcal{E}$ due to deletion of the weight parameters, $\mathbf{w}_i^1$, associated with feature $i$<br>Taylor's Series approximations for both $\mathcal{E}$ and $d\mathcal{E}$ are used, where $d\mathcal{E}$ requires simplifying assumptions |
| $T^r$ | Weight-based saliency [72]<br>$r \neq \infty$ | $\left\lVert \mathbf{w}_i^1 \right\rVert_r$ | N/A |
|  | $r = \infty$ | $\left\lVert \mathbf{w}_i^1 \right\rVert_\infty$ | N/A |

## Table 10. Detailed Notational Definitions and Relationships

| Saliency Metric | Detailed Definition | Relationships |
|---|---|---|
| $\Lambda_i$ | $\dfrac{\int_{\mathcal{R}} \int_{f_i=0}^{f_i=\sum_{h=1}^{K} \left\lvert \delta_h^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 \right\rvert} f_i \, df_i \, d\mathcal{R}}{V_i}$ ,<br>where $V_i$ is defined in Equation 27<br>$\mathcal{R}$ represents the feature space region | |
| $\hat{\Lambda}_i$ | $\sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{h=1}^{K} \left\lvert \delta_h^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 \right\rvert$ | |
| $\hat{\Lambda}_i^{\mathrm{pseudo}}$ | $(PRM)^{-1} \sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{h=1}^{K} \left\lvert \delta_h^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 \right\rvert$ | $\hat{\Lambda}_i^{\mathrm{pseudo}} = (PRM)^{-1} \hat{\Lambda}_i$<br>$\hat{\Lambda}_i^{\mathrm{pseudo}}$ is an approximation to $\Lambda_i$ |
| $\hat{\Lambda}_i^{\mathrm{data}}$ | $P^{-1} \sum_{p=1}^{P} \sum_{h=1}^{K} \left\lvert \delta_h^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 \right\rvert$ | $\hat{\Lambda}_i^{\mathrm{data}}$ is an estimator of $\Lambda_i$ |
| $\Omega_i$ | $\sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{h=1}^{K} \left\lvert \delta_h^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 \right\rvert$ | $\Omega_i = \hat{\Lambda}_i$<br>Equation 50 |
| $\hat{\Omega}_i$ | $\sum_{p=1}^{P} \sum_{m=1}^{M} \sum_{r=1}^{R} \sum_{h=1}^{K} \sum_{l \neq h}^{K} \left\lvert \delta_l^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jl}^2 \hat{w}_{ij}^1 \right\rvert$ | $\hat{\Omega}_i = (K-1)\hat{\Lambda}_i$<br>Equation 46<br>$\hat{\Omega}_i \geq \Omega_i$<br>Equation 47 |
| $\Gamma_i$ | $P^{-1} \sum_{p=1}^{P} \left\lvert \delta_{h_{\max}}^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jh_{\max}}^2 \hat{w}_{ij}^1 \right\rvert$ | $\Gamma_i = \hat{\Lambda}_i^{\mathrm{data}} - P^{-1} \sum_{p=1}^{P} \sum_{h \neq h_{\max}}^{K} \left\lvert \delta_h^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 \right\rvert$<br>$\Gamma_i = \frac{1}{2} \hat{\Lambda}_i^{\mathrm{data}}$ for $K = 2$<br>$\Gamma_i \leq \frac{1}{2} \hat{\Lambda}_i^{\mathrm{data}}$ for $K > 2$<br>Equations 55 and 56 |
| $\rho_i$ | $P^{-1} \sum_{p=1}^{P} \sum_{h=1}^{K} \left\lvert \delta_h^2 \sum_{j=1}^{H} \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 x_i \right\rvert$<br>see Appendix A for derivation | This metric amounts to weighting the saliency, $\hat{\Lambda}_i^{\mathrm{data}}$, for each vector by the feature $x_i$<br>Note: if the relevance factor were associated with the vector of weights, $\hat{w}_i^1$, connected to feature $i$, then the result would be: $P^{-1} \sum_{p=1}^{P} \sum_{h=1}^{K} \sum_{j=1}^{H} \left\lvert \delta_h^2 \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 x_i \right\rvert$, which is very close to $s_i$ |
| $s_i$ | $P^{-1} \sum_{p=1}^{P} \sum_{h=1}^{K} \sum_{j=1}^{H} [\delta_h^2 \delta_j^1 \hat{w}_{jh}^2 \hat{w}_{ij}^1 x_i]^2$<br>see Appendix A for derivation | |
| $\Upsilon^r$<br>$r \neq \infty$<br>$r = \infty$ | $(\sum_{j=1}^{H} \left\lvert \hat{w}_{ij}^1 \right\rvert^r)^{\frac{1}{r}}$<br>$\max\left\{ \lvert \hat{w}_{i1}^1 \rvert, \cdots, \lvert \hat{w}_{iH} \rvert \right\}$ | $\hat{\Lambda}_i^{\mathrm{data}} \leq \Phi' \left\lvert \hat{w}_i^1 \right\rvert$ and $\Upsilon^r \equiv \left\lVert \hat{w}_i^1 \right\rVert_r$<br>Equation 39 |

Table 11. Description of FLIR Features Evaluated

| Feature Number | Feature | Description |
|---|---|---|
| 1 | Length/Width | Ratio of object length to width |
| 2 | Standard Deviation | Standard deviation of pixel values on object |
| 3 | Maximum Brightness | Maximum brightness on object |
| 4 | Compactness | Ratio of number of pixels on object to number of pixels in rectangle which bounds object |
| 5 | Complexity | Ratio of border pixel to total object pixels |
| 6 | Mean Contrast | Contrast ratio of object's mean to local background mean |
| 7 | Contrast Ratio | Contrast ratio of object's highest pixel to its lowest |
| 8 | Bright Pixel Ratio | Ratio of number of pixels on object within 10% of maximum brightness to total object pixels |
| 9 | Difference of Means | Difference of object and local background means |

Adapted from Ruck [60:42]

eliminated one-by-one based on the saliency metric rankings. Finally, the set of saliency metrics are factor analyzed to look for any underlying statistical relationships among the different metrics.

Saliency results are summarized for 30 'trained' neural networks which were trained with the same data set, but with different random initial weights and a different random order of training vector presentation. A data set of 550 vectors was randomly partitioned for each neural network into training, training-test, and validation sets of size 300, 125, and 125, respectively. The neural networks were trained with four middle nodes for 500 epochs before the saliency was computed. From some pilot runs, four middle nodes seemed to be a minimal network structure for this data, since results were degraded for fewer middle nodes and no significant improvements were realized with additional middle nodes. Log-linear declining learning rates and a momentum rate of 0.30 were used to improve the neural network's convergence to a local minimum. Also, a seven percent minimum training set classification error was required for the network's solution to be considered from a 'trained' network. Thirty networks were trained, and all 30 networks met the seven percent requirement. The average training and training-test set classification errors for the 30 neural networks was 3.30 and 9.39 percent, respectively.

In Table 12, rankings are shown for the saliency metrics, where a ranking of '1' indicates the best feature and a ranking of '9' indicates the worst feature. With the exception of the metrics $\rho_i$ and $\bar{s}_i$, similar features receive high rankings, and similar variables receive low rankings across the metrics.

The mean saliency and the corresponding standard error for the 30 runs are presented in Tables 13 and 14. Some further analysis is done using 'saliency function' ratios to evaluate the relative saliency of one feature to another.

Differences in the mean value of the derivative-based metrics presented in Section 3.2 are due to the method used to sample the saliency function $\sum_{k=1}^{K} |\frac{\partial z_k}{\partial x_i}|$ when measuring the saliency for the $i$th feature. The relative saliencies of the feature ranked first to the feature ranked last

Table 12. FLIR Problem: Saliency Metric Rank for 30 Runs

| Feature | $\Lambda_i^{pseudo}$<br>$R = 10$ | $\Lambda_i^{random}$<br>$R = 10$ | $\Lambda_i^{data}$ | $\Gamma_i$ | $\rho_i$ | $s_i$ | $\Upsilon_i^1$ | $\Upsilon_i^2$ | $\Upsilon_i^\infty$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 6 |
| 2 | 2 | 1 | 1 | 1 | 2 | 4 | 2 | 2 | 3 |
| 3 | 9 | 9 | 9 | 9 | 8 | 8 | 9 | 9 | 9 |
| 4 | 8 | 8 | 8 | 8 | 6 | 6 | 8 | 8 | 8 |
| 5 | 1 | 2 | 2 | 2 | 4 | 3 | 1 | 1 | 1 |
| 6 | 5 | 4 | 5 | 5 | 3 | 2 | 3 | 3 | 2 |
| 7 | 4 | 5 | 4 | 4 | 1 | 1 | 6 | 6 | 7 |
| 8 | 3 | 3 | 3 | 3 | 9 | 9 | 5 | 4 | 4 |
| 9 | 6 | 6 | 6 | 6 | 5 | 5 | 4 | 5 | 5 |
| ranked from best to worst | | | | | | | | | |

Table 13. FLIR Problem: Saliency Metric Mean for 30 Runs

| Feature | $\Lambda_i^{pseudo}$<br>$R = 10$ | $\Lambda_i^{random}$<br>$R = 10$ | $\Lambda_i^{data}$ | $\Gamma_i$ | $\rho_i$ | $s_i$ | $\Upsilon_i^1$ | $\Upsilon_i^2$ | $\Upsilon_i^\infty$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.8601 | 0.4939 | 0.6777 | 0.3388 | 0.1652 | 0.7055E-01 | 15.48 | 10.08 | 8.895 |
| 2 | 1.664 | 0.9762 | 1.404 | 0.7018 | 0.4916 | 0.3577 | 22.76 | 14.62 | 11.98 |
| 3 | 0.3663 | 0.2142 | 0.3096 | 0.1549 | 0.1430 | 0.5637E-01 | 7.174 | 4.234 | 3.401 |
| 4 | 0.5866 | 0.3909 | 0.4379 | 0.2189 | 0.2127 | 0.9743E-01 | 11.66 | 7.453 | 6.357 |
| 5 | 1.664 | 0.9318 | 1.358 | 0.6789 | 0.4484 | 0.4564 | 28.27 | 18.57 | 14.29 |
| 6 | 1.136 | 0.7141 | 0.8661 | 0.4330 | 0.4484 | 0.5341 | 20.71 | 14.34 | 13.06 |
| 7 | 1.157 | 0.6786 | 0.9653 | 0.4826 | 0.6739 | 0.5827 | 16.40 | 10.42 | 8.343 |
| 8 | 1.342 | 0.7608 | 1.174 | 0.5866 | 0.6516E-01 | 0.9747E-02 | 17.84 | 12.08 | 10.19 |
| 9 | 1.012 | 0.5840 | 0.8272 | 0.4135 | 0.2896 | 0.1889 | 17.85 | 11.35 | 9.030 |
| bias | 0.7502 | 0.4782 | 0.6013 | 0.3006 | 0.6013 | 0.5586 | 12.27 | 8.106 | 6.992 |

Table 14. FLIR Problem: Saliency Metric Standard Deviation for 30 Runs

| Feature | $\Lambda_i^{pseudo}$<br>$R = 10$ | $\Lambda_i^{random}$<br>$R = 10$ | $\Lambda_i^{data}$ | $\Gamma_i$ | $\rho_i$ | $s_i$ | $\Upsilon_i^1$ | $\Upsilon_i^2$ | $\Upsilon_i^\infty$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.829E-01 | 0.625E-01 | 0.529E-01 | 0.265E-01 | 0.119E-01 | 0.111E-01 | 1.65 | 1.04 | 0.932 |
| 2 | 0.702E-01 | 0.581E-01 | 0.850E-01 | 0.425E-01 | 0.331E-01 | 0.534E-01 | 0.886 | 0.604 | 0.588 |
| 3 | 0.392E-01 | 0.243E-01 | 0.429E-01 | 0.214E-01 | 0.197E-01 | 0.147E-01 | 0.620 | 0.368 | 0.323 |
| 4 | 0.324E-01 | 0.347E-01 | 0.298E-01 | 0.148E-01 | 0.144E-01 | 0.157E-01 | 0.666 | 0.430 | 0.401 |
| 5 | 0.663E-01 | 0.459E-01 | 0.884E-01 | 0.442E-01 | 0.299E-01 | 0.534E-01 | 1.06 | 0.605 | 0.401 |
| 6 | 0.525E-01 | 0.427E-01 | 0.647E-01 | 0.324E-01 | 0.316E-01 | 0.641E-01 | 0.817 | 0.422 | 0.372 |
| 7 | 0.420E-01 | 0.369E-01 | 0.540E-01 | 0.270E-01 | 0.379E-01 | 0.470E-01 | 0.564 | 0.341 | 0.387 |
| 8 | 0.959E-01 | 0.334E-01 | 0.121 | 0.603E-01 | 0.612E-02 | 0.162E-02 | 0.887 | 0.649 | 0.636 |
| 9 | 0.406E-01 | 0.310E-01 | 0.541E-01 | 0.270E-01 | 0.189E-01 | 0.201E-01 | 0.495 | 0.373 | 0.375 |
| bias | 0.443E-01 | 0.275E-01 | 0.501E-01 | 0.251E-01 | 0.501E-01 | 0.637E-01 | 0.611 | 0.342 | 0.308 |

using rankings from $\hat{\Lambda}_i^{\text{data}}$ are 4.54, 4.56, and 4.53, respectively for $\hat{\Lambda}_i^{\text{pseudo}}$, $\hat{\Lambda}_i^{\text{random}}$, and $\hat{\Lambda}_i^{\text{data}}$. As expected, the metric $\Gamma_i$ (to within roundoff error) is exactly equal to its upper bound of $\frac{1}{2}\hat{\Lambda}_i^{\text{data}}$, since this is a two class problem. Despite differences in sampling and the higher dimensional feature space ($M = 10$), the derivative-based saliencies all perform quite similarly.

Among themselves, the weight saliencies have rankings and 'saliency function' ratios which are similar to each other. The relative saliency of the feature ranked first to the feature ranked last using $\hat{\Lambda}_i^{\text{data}}$'s rankings are: 3.17, 3.45, and 3.52 for the three weight saliencies $\Upsilon_i^1$, $\Upsilon_i^2$, and $\Upsilon_i^\infty$. The similarity in the weight-based saliency rankings and 'saliency function' ratios indicates that the choice of $r$ for weight saliency $\Upsilon_i^r$ makes no appreciable difference.

As seen with the examples presented in Section 3.3.3, the 'saliency function' ratios associated with the weight-based saliencies are smaller than those experienced with the derivative-based saliencies. However, a minimal number of nodes were used to analyze the FLIR problem, these results indicate that the relative saliencies of important to irrelevant features are degraded when using the weight-based saliency metrics for greater than one middle node. This phenomena is also associated documented for redundant middle nodes in Section 3.3.3.

The metrics $\rho_i$ and $\bar{s}_i$ have similar rankings to each other (six of the nine features are ranked the same), but not to the other saliency metrics documented. It is interesting to look at the relative saliency of the features ranked first and last by both $\rho_i$ and $\bar{s}_i$. In this case, the respective relative saliencies are 10.34 and 59.78, which are markedly different. The corresponding relative saliencies for the derivative-based metrics are 0.86, 0.89, and 0.82, and the corresponding relative saliencies for the weight-based metrics are 0.92, 0.86, and 0.82. These results indicate that the metrics $\rho_i$ and $\bar{s}_i$ are fundamentally different from the other saliency metrics.

The saliency feature rankings from Table 12 are used to perform systematic feature elimination and evaluation of the network's corresponding classification accuracy. The features are eliminated one-by-one based on the saliency metric rankings, i.e. the worst features are removed first. The results from this analysis are shown in Figure 10, where the $x$-axis corresponds to the number of features which have been removed, and the $y$-axis corresponds to the network accuracy. Four middle nodes are used for the entire evaluation, although fewer middle nodes might have been more appropriate as more features were eliminated. A log-linear declining learning rate was also employed as before. For each neural network, the data set is partitioned as before into training, training-test and validation sets of 300, 125 and 125 vectors, respectively. The data-base partitioning and confidence interval procedures described in Chapter I are used to compute the network accuracy Since the error rate of the training-test set no longer exhibits wide variations after approximate 100 training epochs, 150 epochs were used for each neural network.

In Figure 10, the mean accuracy and a 95 percent confidence interval error band (plotted as horizontal bars about the means) are plotted using the best 10 of 30 neural networks for $\hat{\Lambda}_i^{\mathrm{data}}$. Only a subset of the 'best' neural networks are used to compute the means and standard deviations, since backpropagation learning may not converge to a local minima (see discussion at the end of Section 3 of Chapter I) [80:143]. Although only one of the nine metrics is represented in Figure 10, it is representative of the other metrics. Identical features are retained for many of the saliency metrics at each point in the analysis, so the results from the other metrics are similar.

The results from all of the evaluated metrics indicate it is possible to remove one to three features based on saliency metric rankings with little or no degradation in the average network error. However, further reduction of the feature set requires a trade-off in classification error. This is demonstrated in Figure 10 for $\hat{\Lambda}_i^{\mathrm{data}}$. When more than three features are eliminated, there is great variation in the average network performance depending on which saliency metric was used. This is not surprising, since the saliency of the features was measured when all the features were in the network. If the feature saliencies were re-evaluated for the smaller subset of features, the remaining features would be ranked differently in many cases. Nevertheless, the significant point is that all of the metrics are able to rank expendable features last.

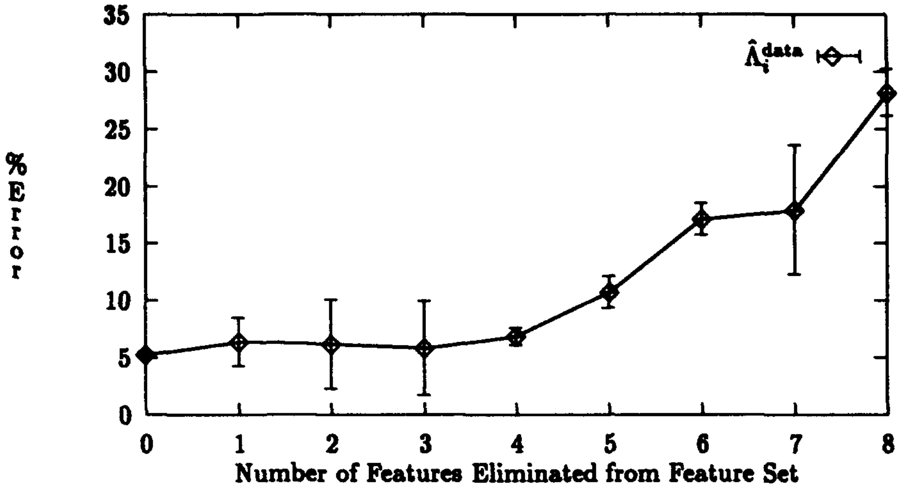


Figure 10. Average Validation Set Accuracy on FLIR Problem as Features are Eliminated

An exploratory factor analysis using a varimax rotation (see Dillon and Goldstein for details [15]) is done on the correlation matrix of the various saliency metrics. The exploratory factor analysis was done in hopes of identifying whether some of the saliency metrics are statistically related to each other by some underlying factor. Using the saliency results from thirty neural network simulations, a factor analysis was performed for each feature in the FLIR problem. The results are displayed in Table 15. In all cases, a two factor model was most appropriate.

For each feature, all of the saliency metrics, generally, load together on the first factor to explain between 70 and 94 percent of the variance in the saliency prior to the varimax rotation. From Table 15, it can be seen that $\hat{\Lambda}_i^{\text{data}}$, $\Gamma_i$, $\rho_i$, and $\bar{s}_i$ have a strong statistical relationship for this problem, since they consistently load on the same factor for each feature. There is also a statistical relationship between $\hat{\Lambda}_i^{random}$ and the weight-based saliencies, $\Upsilon_i^1$, $\Upsilon_i^2$, and $\Upsilon_i^\infty$. The metric $\hat{\Lambda}_i^{\text{pseudo}}$ did not consistently load with one factor or the other; however, seven out of nine times it loaded with $\hat{\Lambda}_i^{\text{data}}$.

Table 15. FLIR Problem: Saliency Metric Loadings after Varimax Rotation

| | Features | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Saliency Metrics | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $\hat{\Lambda}_i^{\text{pseudo}}$ | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |
| $\hat{\Lambda}_i^{\text{random}}$ | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\hat{\Lambda}_i^{\text{data}}$ | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\Gamma_i$ | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\rho_i$ | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\bar{s}_i$ | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\Upsilon_i^1$ | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\Upsilon_i^2$ | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\Upsilon_i^\infty$ | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

Each column corresponds to a factor analysis performed on the saliency metrics for the feature in that column

The numbers indicate which factor a saliency metric loaded on after a varimax rotation

It is interesting to note that factor analysis results for all the features indicate a statistical relationship between $\rho_i$, $\bar{s}_i$, and the metrics $\Gamma_i$ and $\hat{\Lambda}_i^{data}$ for this problem, since they correspond to markedly different saliency rankings and 'saliency function' ratios. This statistical relationship is, most likely, due to the fact that all of these metrics are evaluated only at the known data points. This may indicate an underlying 'sampling factor.' It is also true that the terms in $\rho_i$ are weighted averages of the terms in $\hat{\Lambda}_i^{data}$, where the weighting term for the $p$th exemplar is $x_i^p$. Other similarities between $\rho_i$ and $\bar{s}_i$ are discussed in Appendix A.

*3.5.4 Summary* The relationships between the available neural network feature saliency metrics are documented in this section. The definitions and relationships presented in Tables 9 and 10 consolidate what is known about the available feature saliency metrics.

A 'real world' problem is analyzed in this section to evaluate the set of available feature saliency metrics including those introduced in Section 3.2 and 3.4 of this chapter. The derivative-based metrics discussed in Section 3.2 all have similar rankings and 'saliency function' ratios. The weight-based metrics all have similar rankings and 'saliency function' ratios. The weight-based metrics had rankings similar to the derivative-based metrics, but the 'saliency function' ratios between important and unimportant features are smaller for the weight-based saliency metrics.

The metrics $\rho_i$ and $\bar{s}_i$ are statistically related to each other and the derivative-based metrics by an underlying 'sampling factor,' since all of these metrics are evaluated using the known data. However, their saliency metric rankings are markedly different from the other metrics. Also, their 'saliency function' ratios for important to unimportant features are not similar to each other or to the other metrics. This indicates that the results from $\rho_i$ and $\bar{s}_i$ are fundamentally different from the other saliency metrics despite the underlying 'sampling factor.'

One final observation is made from the feature-by-feature elimination using the saliency metric rankings. All of the metrics, despite fundamental differences, had a set of 'worst' ranked features which did not drastically affect the classification accuracy when removed from the feature set (see Figure 10).

### 3.6 Summary

There are four important research results presented in this chapter. These results are summarized in the next four paragraphs followed by a discussion of which metrics are recommended.

In Section 2, a framework is developed and used for analyzing a variety of derivative-based metrics. Sampling modifications for an established metric are evaluated in this framework. Using 'saliency function' ratios, the saliency metric sensitivities to sampling, training, and redundant middle nodes are evaluated. The metrics do not appear to be particularly sensitive to sampling; however, they are sensitive to redundant middle nodes and the amount of training. It is most important to eliminate redundant middle nodes, since the metrics are most sensitive to training effects in the presence of redundant middle nodes. Increased training may cause the weights associated with the redundant nodes to grow disproportionately. This can contaminate saliency results, since the weights from irrelevant features can get large.

In Section 3, a theoretical relationship is shown between derivative-based and weight-based saliency. In summary, the derivative-based feature saliency metrics are bounded above by a constant linear combination of the feature weights. At one middle node, the 'saliency function' ratios produced with the weight-based metrics and derivative-based metrics are equal, to within roundoff error. This occurs because the constant term directly cancels for the 'saliency function' ratio between any two features. When additional middle nodes are used, empirical results indicate that the relative saliency of important to unimportant features is smaller with weight-based saliency than it is for derivative-based saliency. For problems with redundant middle nodes, this is partly due to the growth of the irrelevant weights associated with redundant middle nodes.

In Section 4, contributions are made in the area of Bayesian-based feature saliency. First, a succinct and exact relationship is demonstrated between a previously suggested Bayesian-based metric and derivative-based saliency. Then a novel Bayesian-based saliency metric using the partial derivative of classifier error is introduced. The computation of this metric requires only a subset of the terms associated with the previously suggested Bayesian-based metric. Finally, the relationship between the new Bayesian-based saliency and derivative-based saliency is derived, and an upper bound for the Bayesian-based saliency is defined. For a two class problem, the new metric produces results exactly equivalent to derivative-based saliency.

In Section 5, a catalogue of feature saliency metric definitions and relationships is presented in Tables 9 and 10. In this section, the catalogue of metrics are evaluated for a 'real world' problem. On this problem, saliency rankings, 'saliency function' ratios, and factor analysis are used to empirically evaluate similarities and differences between the saliency metrics. One similarity is that, despite differences, all of the metrics consistently ranked a set of 'nonessential' features last.

Since metrics perform differently, the remainder of this section summarizes recommendations for selecting a feature saliency metric.

For discriminant analysis problems using networks with more than one middle node, the saliency metric $\Gamma_i$ in Equation 53 on page 86 is preferable to $\hat{\Lambda}_i^{\text{data}}$ in Equation 31 on page 68 for two reasons. First, it is intuitively appealing, since it represents a saliency metric which is related to the average classifier $P_e$. Second, it provides feature rankings using a subset of the terms required for computation of $\hat{\Lambda}_i^{\text{data}}$.

For function approximation or discriminant analysis problems using networks with more than one middle node, the saliency metric $\hat{\Lambda}_i^{\text{data}}$ in Equation 31 on page 68 should be preferred over $\hat{\Lambda}_i$. The saliency metric $\hat{\Lambda}_i^{\text{data}}$ provides good feature rankings, requires less computation than $\hat{\Lambda}_i$, and is based on information known about the data from feature space. Furthermore, the metric $\hat{\Lambda}_i$ requires a tactical decision concerning the amount of 'pseudo-sampling.'

For any classification or function approximation problem using a network with only one middle node, the weight-based metrics $\Upsilon_i^r$ (see Equation 32 on page 78) are best. In this case, the relative saliencies produced using weight-based saliency will be identical to $\Gamma_i$ or $\hat{\Lambda}_i^{\text{data}}$. For networks using more than one middle node, weight-based saliency can still be used for a cursory analysis of a feature's relative importance. However, the empirical results suggest that the relative importance of one feature to another is degraded when additional middle nodes are used. Despite a potential degradation in results, weight-based saliency is still appealing. It is related to the metric $\hat{\Lambda}_i^{\text{data}}$, and it is directly computable from a trained network without reevaluating the data.

The metrics $\hat{\Lambda}_i^{\text{random}}$, $\bar{s}_i$ and $\rho_i$ are not recommended, because they require unnecessary work or additional assumptions with no gains in performance over the other available saliency metrics. The experimental pseudo-sampling metric $\hat{\Lambda}_i^{\text{random}}$ is not preferred, since it invokes unnecessary random sampling and potentially greater computation. The second order metric, $\bar{s}_i$, is not recommended, since it is associated with additional assumptions which are used to simplify the evaluation of the Taylor's series expansion of the network error. The relevance metric, $\hat{\rho}_i$, is not recommended, since it requires the assumption that $\rho_i$ can be approximated using a partial derivative of the error with respect to a relevance factor.

The results shown in Section 3.5 of this chapter indicate that the least important features, as ranked by any of the feature saliency metrics, may not be essential to good classification accuracy. In the next chapter, feature screening techniques are presented which can be used to formally identify unimportant or *noise-like* features.

# *IV. Feature Screening Techniques for Feedforward Neural Networks*

## *4.1 Introduction*

Feature screening can be used to take a preliminary look at a set of features. Specifically, it can be used to identify and eliminate noisy features prior to a formalized feature selection procedure. Two feature screening procedures are discussed and evaluated in this section. The first procedure, presented in Section 2, is based on statistically comparing the saliency of candidate features to the saliency of a noisy feature. The second procedure, presented in Section 3, is based on the weight screening hypothesis test proposed by White [82]. These procedures are summarized and recommendations are made in Section 4.

As in Chapter III, the research and theoretical results presented in this chapter reflect the exclusive use of the sigmoidal activation functions on the middle and output nodes of a feedforward neural network (presented in Chapter I). The fundamental definitions of network output activations, saliency, and network derivatives will change for other types of activation functions. However, the underlying procedures presented in this chapter remain the same. The neural network notational conventions, network structure, and the backpropagation algorithm introduced in Section 3 of Chapter I are used as necessary in this chapter.

## *4.2 Feature Saliency Screening*

*4.2.1 Introduction.* A saliency screening procedure which statistically formalizes the screening methodology proposed by Belue and Bauer is presented in this section [5]. In their procedure, Belue and Bauer augment a set of features with a known irrelevant feature (i.e. noise). They recommend eliminating any feature whose mean saliency falls inside a one-sided confidence interval about the mean saliency of noise. Mean saliency is calculated from the feature saliency results of several trained neural networks.

In the formalized procedure presented here, the entire feature set is simultaneously screened for nonsalient features using a specified degree of statistical confidence. The formalized procedure requires an injected noise feature, a Bonferroni test statistic, and an appropriate hypothesis for testing the equality of means. These are reviewed before the formalized screening procedure is presented. The saliency screening procedure is evaluated on the XOR problem and on two application problems to determine if artificially included noise features can be identified.

The saliency metric $\hat{\Lambda}_i^{\text{data}}$ is used in this discussion, although a different saliency metric could easily have been used. For reference, $\hat{\Lambda}_i^{\text{data}}$ defined in Equation 31 of Chapter III is given again as:

$$\hat{\Lambda}_i^{\text{data}} = P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} \left| \frac{\partial z_k}{\partial x_i}(\mathbf{x}^p, \hat{\mathbf{w}}) \right|$$

*4.2.2 Noise Injection.* There are two reasons for augmenting the feature set with noise. First, the saliency of noise features is significantly greater than zero; therefore, a test cannot be developed based on $\hat{\Lambda}_i^{\text{data}} = 0$. Second, the actual magnitude of this saliency can be different from problem to problem; therefore, the magnitude of *noise-like* features should be characterized for the problem at hand. This entails:

- Augmenting the original feature set with a true noise feature, $x_s$ . One way to generate a noise feature is by drawing random numbers from a Uniform (0,1) distribution.
- Training a neural network using the augmented feature set to minimize the *training-test* set error.
- Calculating the feature saliency for all the features (including the noise).

*4.2.3 Bonferroni Joint Hypothesis Testing.* The Bonferroni procedure accounts for the fact that the significance level for a 'family' of tests is not the same as that for an individual test. To conduct $M$ one-sided hypothesis tests with a 'family' significance level of $\alpha$, each individual test must be conducted with an individual significance level of $\frac{\alpha}{M}$ [47:594]. The $M$ hypothesis tests correspond to the $M$ candidate features in the feature set. The individual significance level is found using the Bonferroni inequality which is derived from simple probability theorems.

Let $A_i$ be the event that the $i$th hypothesis test is rejected when it is true. The probability of $A_i$ is denoted as $P(A_i)$ and the confidence coefficient for $A_i$ be defined $1 - P(A_i)$. The following set of probability theorems are used to derive the Bonferroni inequality.

**Probability Theorem 1**

$$P(A_i \cup A_j) = P(A_i) + P(A_j) - P(A_i \cap A_j)$$

**Probability Theorem 2**

$$P(\bar{A}_i) = 1 - P(A_i)$$

**Probability Theorem 3**

$$P(\overline{A_i \cup A_j}) = P(\bar{A}_i \cap \bar{A}_j)$$

Using these basic probability theorems, the following statement can be derived.

$$P(\bar{A}_i \cap \bar{A}_j) = 1 - P(A_i) - P(A_j) + P(A_i \cap A_j)$$

Since $P(A_i \cap A_j)$ is greater than or equal to zero, the Bonferroni inequality is defined as [47:160]:

$$P(\bar{A}_i \cap \bar{A}_j) \geq 1 - P(A_i) - P(A_j)$$

The right hand side of the Bonferroni inequality represents a conservative estimate of the joint confidence coefficient for the individual hypothesis tests $A_i$ and $A_j$.

The Bonferroni inequality can be generalized for simultaneous hypothesis testing of $M$ individual hypothesis tests denoted $A_1, A_2, \cdots A_M$ as

$$P(\bar{A}_1 \cap \bar{A}_2 \cap \cdots \cap \bar{A}_M) = 1 - (\ P(A_1) + P(A_2) + \cdots + P(A_M)\ )$$

Table 16. Bonferroni Critical Values

| | Individual Significance Levels | | |
|---|---|---|---|
| Degrees of Freedom $\nu$ | $\frac{\alpha}{M} = 0.01$ | $\frac{\alpha}{M} = 0.005$ | $\frac{\alpha}{M} = 0.0005$ |
| 10 | 2.764 | 3.169 | 4.587 |
| 20 | 2.528 | 2.845 | 3.850 |
| 30 | 2.457 | 2.750 | 3.646 |
| 40 | 2.423 | 2.704 | 3.551 |
| 60 | 2.390 | 2.660 | 3.460 |
| 120 | 2.358 | 2.617 | 3.373 |
| $\infty$ | 2.326 | 2.576 | 3.291 |

$\alpha$: 'family' level of significance, where $\alpha = M \cdot (\frac{\alpha}{M})$

$M$ is the number of individual hypothesis tests

Now let $P(A_i) = \beta$ for each of the $M$ individual hypothesis tests. The result is that the joint hypothesis test has a confidence coefficient of $1 - M\beta$. For the joint test to have a confidence coefficient of $1 - \alpha$ (or a 'family' significance level of $\alpha$), the individual hypothesis tests must have significance levels of $\beta$ equal to $\frac{\alpha}{M}$.

To produce a 'family' significance level of $\alpha$, the Bonferroni critical value, denoted $B$, is used for each individ'.al hypothesis test. For $M$ one-sided tests, the Bonferroni critical value is:

$$B = t_{1-\frac{\alpha}{M};\nu} \tag{57}$$

where $\nu = N - 1$ and $N$ is the number of observations on which the tests are based. Table 16 contains Bonferroni critical values $B$ for combinations of $\nu$ and $\frac{\alpha}{M}$.

*4.2.4 Hypothesis Tests for the Equality of Means.* The saliency screening methodology consists of simultaneously applying individual hypothesis tests on the equality of two means: (1) the mean saliency of feature $i$, denoted $\mu_{\hat{\Lambda}_i^{\text{data}}}$, and (2) the mean saliency of noise, denoted $\mu_{\hat{\Lambda}_n^{\text{data}}}$. In general, this type of hypothesis test requires two conditions:

1. The saliency for each feature $i$ should be a normally distributed random variable, or if the saliency is nonnormal, the conditions of the central limit theorem (CLT) should hold [24:280].
2. The saliency for the $i$th feature is independently distributed with mean $\mu_{\hat{\Lambda}_i^{\text{data}}}$ and variance $\sigma^2_{\hat{\Lambda}_i^{\text{data}}}$. Similarly, the saliency for the augmented noise feature is independently distributed with mean $\mu_{\hat{\Lambda}_n^{\text{data}}}$ and variance $\sigma^2_{\hat{\Lambda}_n^{\text{data}}}$ [24:281].

For the saliency screening methodology, the first condition must be met by using the CLT, since the theoretical distribution of $\hat{\Lambda}_i^{\text{data}}$ has not been shown to be normal. The CLT states [47:6]:

**Theorem 1** *If $Y_1, \cdots, Y_N$ are independent random observations from a population with probability function $f(Y)$ for which $\sigma^2\{Y\}$ is finite, the sample mean $\bar{Y}$:*

$$\bar{Y} = \frac{\sum_{j=1}^{N} Y_n}{N}$$

*is approximately normally distributed when the sample size $N$ is reasonably large, with mean $E\{Y\}$ and variance $\frac{\sigma^2\{Y\}}{N}$.*

To invoke the CLT, the saliency metric $\hat{\Lambda}_i^{\text{data}}$ is defined as

$$\hat{\Lambda}_i^{\text{data}} = \frac{\sum_{p=1}^{P} Y_i^p}{P}$$

where

$$Y_i^p = \sum_{k=1}^{K} \left| \frac{\partial z_k}{\partial x_i}(\mathbf{x}^p, \mathbf{w}) \right|$$

The superscript $p$ corresponds to the $p$th exemplar from a set of $P$ exemplars. The variable $Y_i^p$ is a random observation from a population with probability function $f(Y_i)$ for which $\sigma^2\{Y_i\}$ is finite. The variable $Y_i^p$ is a random observation since it is a function of $\mathbf{x}^p$ which is a random observation from the population of input exemplars. The conditions of the CLT can be applied in a similar fashion for many of the metrics shown in Table 9 in Chapter III.

The second condition for 'equality of means' hypothesis testing is also met. The samples of saliency for each feature are associated with independent realizations of neural networks. Each realization is independent for three reasons: (1) the random partitioning of the data set, (2) the random initialization of the weight parameters, and (3) the random order for presentation of the training data.

The test statistic and the hypothesis test used to test for the equality of two means is defined by one of four cases [24:280-292]. They are:

1. The variances ($\sigma^2_{\hat{\Lambda}_i^{\text{data}}}$ and $\sigma^2_{\hat{\Lambda}_n^{\text{data}}}$) corresponding to the distributions of $\mu_{\hat{\Lambda}_i^{\text{data}}}$ and $\mu_{\hat{\Lambda}_n^{\text{data}}}$ are known. The observations of $\hat{\Lambda}_i^{\text{data}}$ and $\hat{\Lambda}_n^{\text{data}}$ are independent.

2. The variances ($\sigma^2_{\hat{\Lambda}_i^{\text{data}}} = \sigma^2_{\hat{\Lambda}_n^{\text{data}}}$) corresponding to the distributions of $\hat{\Lambda}_i^{\text{data}}$ and $\hat{\Lambda}_n^{\text{data}}$ are unknown but equal. The observations of $\hat{\Lambda}_i^{\text{data}}$ and $\hat{\Lambda}_n^{\text{data}}$ are independent.

3. The variances ($\sigma^2_{\hat{\Lambda}_i^{\text{data}}} \neq \sigma^2_{\hat{\Lambda}_n^{\text{data}}}$) for the distributions of $\hat{\Lambda}_i^{\text{data}}$ and $\hat{\Lambda}_n^{\text{data}}$ are unknown and unequal. The observations of $\hat{\Lambda}_i^{\text{data}}$ and $\hat{\Lambda}_n^{\text{data}}$ are independent.

4. The observations of $\hat{\Lambda}_i^{\text{data}}$ and $\hat{\Lambda}_n^{\text{data}}$ are paired and dependent.

Cases 1, 2, and 3, which each correspond to a null hypothesis of $\mathbf{H}_0 : \mu_{\hat{\Lambda}_i^{\text{data}}} = \mu_{\hat{\Lambda}_n^{\text{data}}}$, are inappropriate, because in this application the observations of $\hat{\Lambda}_i^{\text{data}}$ and $\hat{\Lambda}_n^{\text{data}}$ are paired and dependent. The appropriate hypothesis testing procedure is the Paired $t$-test defined using Case 4. The Paired $t$-test is defined here as:

$$\begin{array}{rll} \text{Null Hypothesis} & \mathbf{H}_0 : & \mu_{D_i} = 0 \\ \text{Alternative Hypothesis} & \mathbf{H}_A : & \mu_{D_i} > 0 \,, \end{array}$$

where $\mu_{D_i}$, the difference between the $i$th feature's mean saliency and the noise feature's mean saliency, is defined

$$\mu_{D_i} = \mu_{\hat{\Lambda}_i^{\mathrm{data}}} - \mu_{\hat{\Lambda}_{\mathrm{n}}^{\mathrm{data}}}$$

Define the test statistic $t^*$ as

$$t^* = \frac{\bar{D}_i}{S_{\bar{D}_i}} \,, \tag{58}$$

where

$$\bar{D}_i = \frac{\sum_{j=1}^{N} D_{i,j}}{N} \tag{59}$$

$$D_{i,j} = \hat{\Lambda}_{i,j}^{\mathrm{data}} - \hat{\Lambda}_{\mathrm{n},j}^{\mathrm{data}}$$

$$S_{\bar{D}_i}^2 = \frac{\sum_{j=1}^{N} (D_{i,j} - \bar{D}_i)^2}{(N-1)N} \,, \tag{60}$$

and $j$ indicates the $j$th of the $N$ samples of $D_{i,j}$. For feature $i$, $\bar{D}_i$ and $S_{\bar{D}_i}^2$ are the sample mean and sample variance, respectively, for $N$ samples. The critical value for an individual significance level of $\alpha$ is $t_{\mathrm{crit}} = t_{1-\alpha,\nu}$, where $\nu = N - 1$. The null hypothesis is rejected if the test statistic, $t^*$, exceeds the critical value $t_{\mathrm{crit}}$.

*4.2.5 Methodology.* The formalized statistical screening procedure for identifying nonsalient features is based on applying the Bonferroni inequality to $M$ individual hypothesis tests to achieve a predetermined 'family' significance level $\alpha$. The individual hypothesis test used is

$$\begin{array}{lll} \textbf{Null Hypothesis} & \mathbf{H}_0: & \mu_{D_i} = 0 \\ \textbf{Alternative Hypothesis} & \mathbf{H}_A: & \mu_{D_i} > 0\ , \end{array}$$

The procedure is summarised as:

### Saliency Screening Procedure

1. Augment feature set with a noise feature, $x_n$ .
2. Train neural net to minimize *training-test* set error.

   All nets should ideally use a minimal network structure with no redundant middle nodes or features.

   All nets should ideally converge to a local minimum and not a saddle point.
3. Compute the feature saliency, $\hat{\Lambda}_i^{\text{data}}$ for each of the features, including $x_n$ .
4. Repeat steps 2 and 3 a minimum of ten times ($N = 10$), using random initialization of weight parameters and random data set partitioning.
5. Select 'family' significance level, $\alpha$.
6. For each feature do an individual hypothesis test as follows:

   (a) Compute $\bar{D}_i$ and $S^2_{\bar{D}_i}$ using Equations 59 and 60 on page 113.

   (b) Compute the test statistic $t^*$ using Equation 58 on page 113.

   (c) Determine the Bonferroni critical value $B = t_{\frac{\alpha}{M},\nu}$.

   (d) Evaluate the test statistic as follows:

   - If $t^* \leq B$, the null hypothesis can not be rejected for feature $i$.

     Conclusion: feature $i$ is nonsalient, since the difference between the $i$th feature's saliency and the noise feature's saliency is not statistically different from zero at the $\alpha$ 'family' significance level.

- If $t^* > B$, reject the null hypothesis for feature $i$.

  Conclusion: feature $i$ is salient, since there is a statistical difference at the $\alpha$ 'family' significance level between the saliency of the $i$th feature and the saliency of the noise feature.

7. Eliminate the nonsalient features and retrain the network with only the salient features.

If there are any questions concerning the salient features, the procedure can be performed again on the reduced feature set to confirm the remaining features are all still salient.

The identification of nonsalient features using saliency screening is related to the sample size. A larger sample size usually corresponds to a smaller standard error for $S^2_{\bar{D}_i}$ and a smaller standard error corresponds to a larger test statistic for $t^*$ in Equation 58. Additionally, a larger sample means a smaller $t$-statistic or Bonferroni critical value $B$. For a feature which is borderline or very nearly *noise-like*, $H_0$ may be more easily rejected with larger sample sizes.

The saliency metrics used in this procedure are sensitive to training and middle nodes as discussed in Chapter III; therefore, this procedure may also be sensitive to training and middle nodes redundancies. These factors are partially minimized for evaluation of the saliency screening procedure, since a minimal number of middle nodes and sufficient training epochs are determined through a series of pilot runs.

*4.2.6 Analysis.* The robustness of the saliency screening methodology is evaluated with three problems. These problems include: the XOR problem, the Armor Piercing Incendiary (API) Projectile Functioning problem, and the FLIR problem. The second and third problems are used to evaluate whether the saliency screening procedure is practical for application problems. For each test problem, the feature set is augmented with one or more additional noise features. Then the saliency screening procedure is used to determine if additional noise features are identified as nonsalient.

*4.2.6.1 XOR Problem.* In this section, the saliency screening procedure is applied to an XOR problem (shown in Figure 9 of Chapter III) which has been augmented with five additional noise features. One of the artificial noise features serves as the augmented noise $x_a$ , and the others serve as candidate features. The procedure is evaluated to see if the candidate noise features are identified during the screening process.

The saliency screening procedure was repeated with $N$ equal to 10, 30, and 100 realizations of 'trained' neural networks which were all trained with the same data set, but with different random initial weights and different training order presentations. A data set of 600 vectors was partitioned into training, training-test, and validation sets of size 400, 100, and 100, respectively. The networks were trained with four middle nodes, which was the minimal network which would reliably train to a local minima. A log-linear declining learning rate and a momentum rate of 0.30 were used. For all runs, 700 epochs were used with a requirement for a seven percent or lower training set classification error for the network's solution to be considered from a 'trained' network.

Saliency screening results for $N = 30$ realizations are reported in Table 17. In total, 38 networks were trained, and 30 networks met the minimum error requirement. The average training and training-test set error rates averaged over the 30 networks were 2.69% and 6.83%, respectively. Using the screening procedure, the artificial noise features are identified as nonsalient, and the features $x$ and $y$ are identified as salient. Equivalent results are produced for the saliency screening procedure with $N = 10$ and $N = 100$ neural network realizations. The other saliency metrics (see Table 9 of Chapter III) also provide similar results when they are used with the saliency screening procedure.

The feature saliency rankings from Table 17 are used for successive removal of candidate features. In Figure 11 the mean error and a 95% confidence interval band (plotted as horizontal

Table 17. Saliency Screening Results on XOR Problem for 30 Runs

| Feature $i$ | $\bar{\hat{\Lambda}}_i^{data}$ Ranking | $\bar{\hat{\Lambda}}_i^{data}$ | $\sigma_{\bar{\hat{\Lambda}}_i^{data}}$ | $\bar{D}_i$ | $S_{\bar{D}_i}$ | $t^* = \frac{\bar{D}_i}{S_{\bar{D}_i}}$ | $\nu$ | Reject $H_0$ |
|---|---|---|---|---|---|---|---|---|
| $x$ | 2 | 2.055 | 0.610E-01 | 1.914 | 0.611E-01 | 31.3 | 29 | yes |
| $y$ | 1 | 2.149 | 0.476E-01 | 2.008 | 0.456E-01 | 44.1 | 29 | yes |
| noise 1 | 6 | 0.133 | 0.148E-01 | -0.796E-02 | 0.148E-01 | -0.536 | 29 | no |
| noise 2 | 7 | 0.128 | 0.111E-01 | -0.132E-01 | 0.840E-02 | -1.570 | 29 | no |
| noise 3 | 4 | 0.142 | 0.145E-01 | -0.893E-03 | 0.142E-01 | 0.630E-01 | 29 | no |
| noise 4 | 3 | 0.151 | 0.148E-02 | -0.104E-01 | 0.179E-01 | 0.582 | 29 | no |
| $x_n$ | 5 | 0.141 | 0.122E-01 | N/A | N/A | N/A | N/A | N/A |

Features ranked from best to worst

Individual Significance Level $\frac{\alpha}{M} = .005$

Bonferroni Critical Value $B = 2.756$

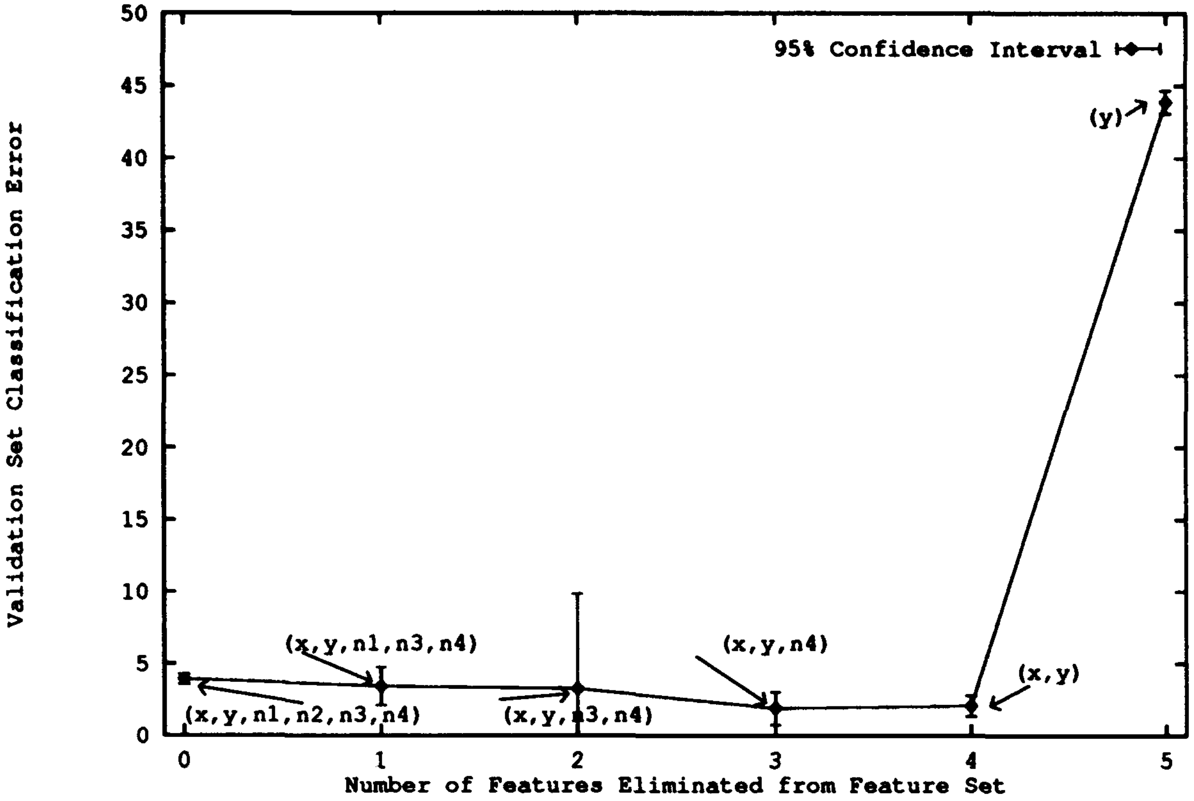


Figure 11. Average Validation Set Error on XOR Problem as Features are Eliminated

bars about the means) are plotted on a validation set of 100 vectors. The mean error at each point is labeled with the features retained in the neural network model. The database partitioning and confidence interval procedures discussed in Chapter I are used to estimate the average network classification errors using the best 30 of 100 realisations of neural network training. The network's accuracy generally improves as the nonsalient candidate features are removed from the training set one-by-one. The 95% confidence interval for the average error is significantly smaller when all the noise has been eliminated from the feature set. Another way of saying this is that prediction variance is reduced after *noise-like* features are eliminated. The network accuracy is only seriously degraded when the first salient feature $x$ is removed.

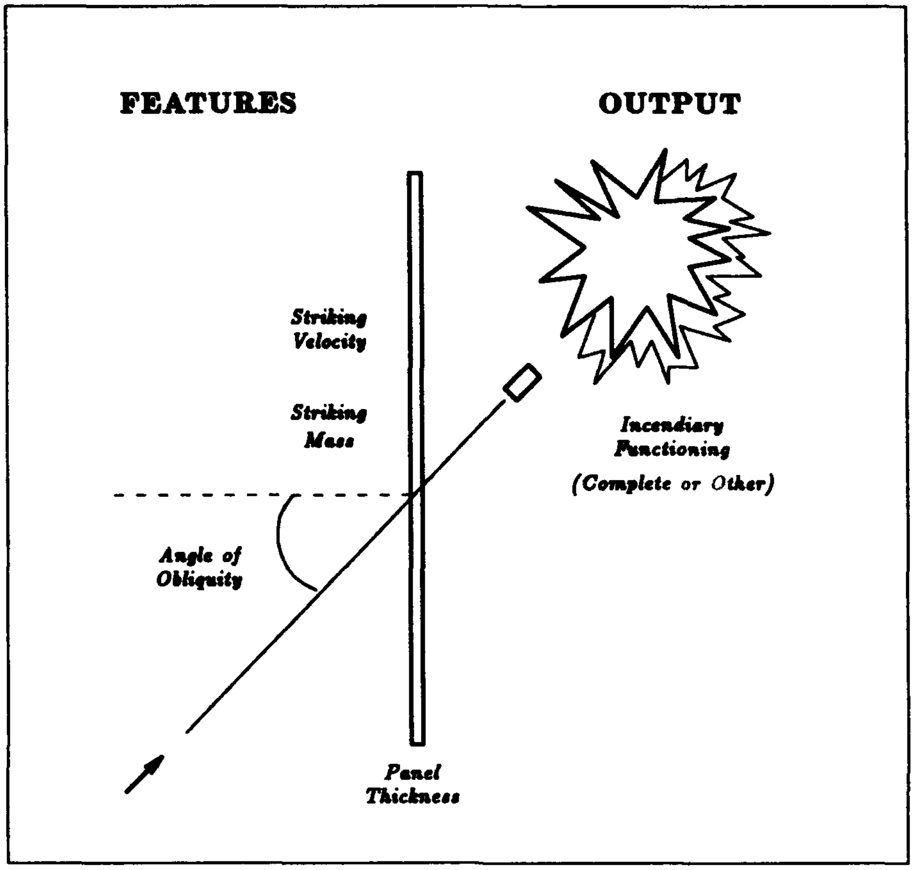


Figure 12. API Projectile Firing

reprinted from Belue [4]

*4.2.6.2 Armor Piercing Incendiary (API) Problem.* In this section, the saliency screening procedure is applied to the API problem which uses data on the performance of API projectiles to classify an API projectile's performance as "complete" or "other [34]." For each shot, four independent parameters are known; impact striking velocity ($V_S$), impact striking mass ($M_S$), panel ply thickness in inches ($PLY$), and the secant of the impact obliquity angle ($SECT$). The firing process is illustrated in Figure 6. A feature vector for this application consists of the parameters of each API projectile firing. The API problem has been augmented with two additional noise features. One of the artificial noise features serves as the augmented noise $x_n$ , and the other serves as a candidate feature. The saliency screening procedure is evaluated to see if the candidate noise feature is identified during the screening process.

The saliency screening procedure was repeated for this application with $N$ equal to 10, 30, and 100 realizations of 'trained' neural networks which were all trained with the same data set, but with different random initial weights and different training order presentations. A data set of 281 vectors was partitioned into training, training-test, and validation sets of size 181, 50, and 50, respectively. The networks were trained with five middle nodes, which was the minimal network which would reliably train to a local minima. A log-linear declining learning rate and a momentum rate of 0.30 were used. For all runs, 700 epochs were used with a requirement for a seven percent or lower training set classification error for the network's solution to be considered from a 'trained' network.

Saliency screening results for $N = 30$ realizations are reported in Table 18. In total, 30 networks were trained, and all 30 networks met the minimum error requirement. The average training and training-test set error rates averaged over the 30 networks were 2.32% and 8.4%, respectively.

Using the screening procedure, the artificial noise feature and the feature $M_S$ are identified as nonsalient. All other features are identified as salient. These results are consistent with Belue and Bauer [7]. Using the saliency screening procedure with $N = 10$ and $N = 100$ neural network realizations produced equivalent results to the results shown in Table 18. Using the saliency screening procedure with the other metrics presented in Table 9 of Chapter III also produced equivalent results.

The feature saliency rankings from Table 18 are used for successive removal of candidate features. In Figure 13 the mean validation error (50 vectors) and a 95% confidence interval band (plotted as horizontal bars about the means) are plotted. The mean error at each point is labeled with the features retained in the neural network model. The database partitioning and confidence

Table 18. Saliency Screening Results on API Problem for 30 Runs

| Feature $i$ | $\bar{\hat{\Lambda}}_i^{\text{data}}$ Ranking | $\bar{\hat{\Lambda}}_i^{\text{data}}$ | $\sigma_{\bar{\hat{\Lambda}}_i^{\text{data}}}$ | $\bar{D}_i$ | $S_{\bar{D}_i}$ | $t^* = \frac{\bar{D}_i}{S_{\bar{D}_i}}$ | $\nu$ | Reject $H_0$ |
|---|---|---|---|---|---|---|---|---|
| $PLY$ | 2 | 0.583 | 0.290E-01 | 0.432 | 0.310E-01 | 13.9 | 29 | yes |
| $V_S$ | 3 | 0.472 | 0.147E-01 | 0.320 | 0.217E-01 | 14.8 | 29 | yes |
| $M_S$ | 5 | 0.150 | 0.119E-01 | -0.939E-03 | 0.179E-01 | -0.525E-01 | 29 | no |
| $SECT$ | 1 | 0.960 | 0.369E-01 | 0.809 | 0.408E-01 | 19.8 | 29 | yes |
| noise | 6 | 0.148 | 0.926E-02 | -0.335E-02 | 0.164E-01 | -0.205 | 29 | no |
| $x_{\text{noise}}$ | 4 | 0.151 | 0.144E-01 | N/A | N/A | N/A | N/A | N/A |

Features ranked from best to worst

Individual Significance Level $\frac{\alpha}{M} = .005$

Bonferroni Critical Value $B = 2.756$

interval procedures discussed in Chapter I are used to estimate the average network classification errors using the best 30 of 100 realizations of neural network training. The 95% confidence interval for the average error is significantly smaller when the noise feature and $M_S$ are eliminated from the features set. Once again, as with the XOR problem, the prediction variance is reduced after *noise-like* features are eliminated. It is only after the next feature $V_S$ is removed that the average network accuracy becomes seriously degraded.

*4.2.6.3 FLIR Problem.* The FLIR problem used in Chapter III is revisited to evaluate the saliency screening procedure. The FLIR feature set described in Table 11 is augmented with two artificial noise features. One of the artificial noise features serves as the augmented noise $x_a$ , and the other serves as a candidate feature. The saliency screening procedure is evaluated to see if the additional noise feature is identified during the screening process.

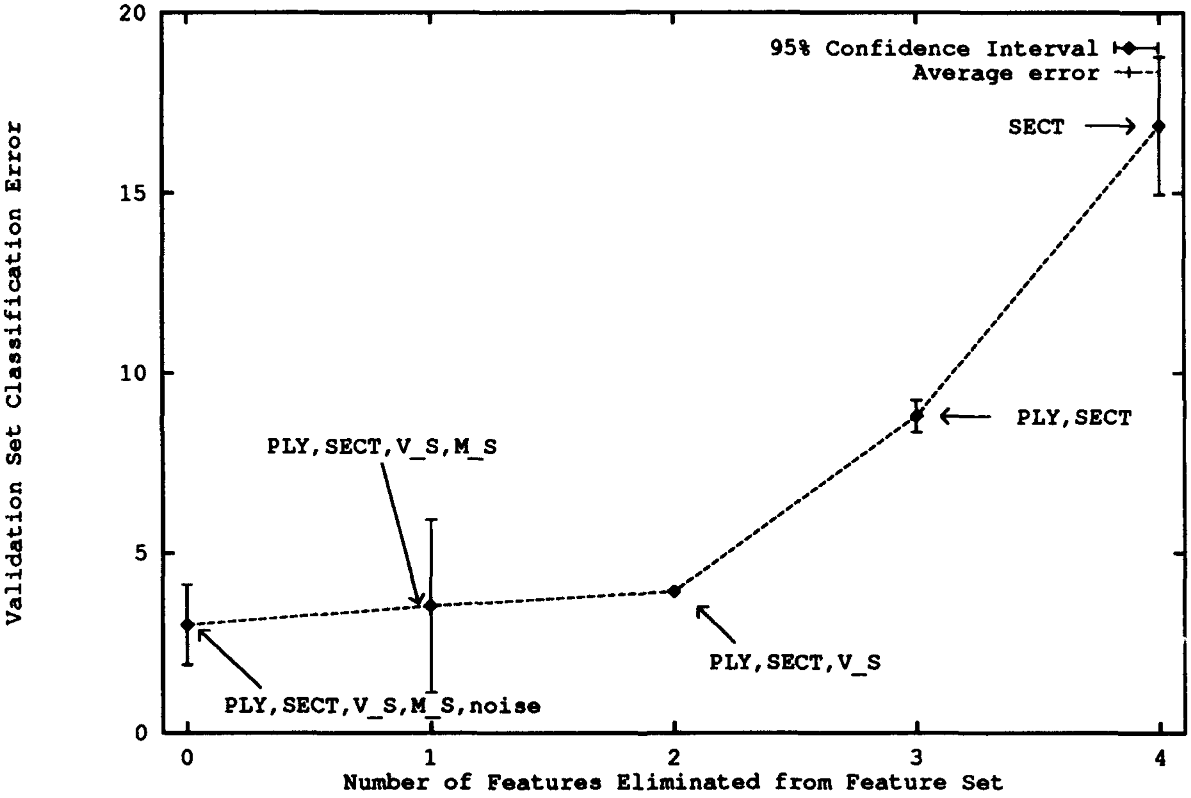


Figure 13. Average Validation Set Error on API Problem as Features are Eliminated

The saliency screening procedure was repeated with $N$ equal to 10, 30, and 100 realizations of 'trained' neural networks which were all trained with the same data set, but with different random initial weights and different training order presentations. A data set of 550 vectors was partitioned into training, training-test, and validation sets of size 350, 100, and 100, respectively. The networks were trained with four middle nodes, which was the minimal network which would reliably train to a local minima. A log-linear declining learning rate and a momentum rate of 0.30 were used. For all runs, 500 epochs were used with a requirement for a seven percent or lower training set classification error for the network's solution to be considered from a 'trained' network.

Saliency screening results for $N = 30$ realizations are reported in Table 19. In total, 30 networks were trained, and all 30 networks met the minimum error requirement. The average training and training-test set error rates averaged over the 30 networks were 3.86% and 8.77%, respectively.

Using the screening procedure for $N = 30$, the artificial noise feature is the only feature identified as nonsalient, and all other features are identified as salient. The third feature 'maximum brightness' has a fairly small test statistic of 5.37 which is only half the size of the next largest test statistic, but it would not be identified as nonsalient with the saliency screening procedure. However, it might warrant further screening after the noise feature is removed.

Using the saliency screening procedure with $N = 100$ neural network realizations produced equivalent results. However, when the saliency screening procedure is used with $N = 10$ neural network realizations the test statistic for 'maximum brightness' drops to 3.02. With an individual significance level $\frac{\alpha}{M} = .005$ it still would not be identified as nonsalient, but at a higher significance level, say $\frac{\alpha}{M} = .0005$, it would be identified. This result demonstrates the relationship between sample size and the identification of nonsalient features using the saliency screening procedure.

When the saliency screening procedure was performed using the other metrics presented in Table 9 of Chapter III, the results are similar. The differences are that feature 3 is identified as

Table 19. Saliency Screening Results on FLIR Problem for 30 Runs

| Feature $i$ | $\bar{\bar{\hat{\Lambda}}}_i^{\text{data}}$ Ranking | $\bar{\bar{\hat{\Lambda}}}_i^{\text{data}}$ | $\sigma_{\bar{\bar{\hat{\Lambda}}}_i^{\text{data}}}$ | $\bar{D}_i$ | $S_{\bar{D}_i}$ | $t^* = \frac{\bar{D}_i}{S_{\bar{D}_i}}$ | $\nu$ | Reject $H_0$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 0.629 | 0.374E-01 | 0.537 | 0.367E-01 | 14.6 | 29 | yes |
| 2 | 1 | 1.537 | 0.564E-01 | 1.444 | 0.572E-01 | 25.2 | 29 | yes |
| 3 | 9 | 0.256 | 0.303E-01 | 0.163 | 0.304E-01 | 5.37 | 29 | yes |
| 4 | 8 | 0.481 | 0.169E-01 | 0.389 | 0.163E-01 | 23.9 | 29 | yes |
| 5 | 2 | 1.447 | 0.506E-01 | 1.355 | 0.499E-01 | 27.1 | 29 | yes |
| 6 | 6 | 0.853 | 0.475E-01 | 0.761 | 0.487E-01 | 15.6 | 29 | yes |
| 7 | 4 | 0.946 | 0.392E-01 | 0.853 | 0.408E-01 | 20.9 | 29 | yes |
| 8 | 3 | 1.208 | 0.941E-01 | 1.115 | 0.939E-01 | 11.9 | 29 | yes |
| 9 | 5 | 0.857 | 0.427E-01 | 0.764 | 0.441E-01 | 17.3 | 29 | yes |
| noise | 10 | 0.115 | 0.106E-01 | 0.229E-01 | 0.141E-01 | 1.62 | 29 | no |
| $x_n$ | 11 | 0.925E-01 | 0.763E-02 | N/A | N/A | N/A | N/A | N/A |

Features ranked from best to worst

Individual Significance Level $\frac{\alpha}{M} = .005$

Bonferroni Critical Value $B = 2.756$

nonsalient for $N = 10$ using the metrics $\hat{\rho}_i$ and $\bar{s}_i$, and feature 8 is identified as nonsalient with the metric $\bar{s}_i$. The conclusion reached is that the results may differ slightly depending on the saliency metric used.

The feature saliency rankings from Table 19 are used for successive removal of candidate features. In Figure 14 the mean error and a 95% confidence interval band (plotted as horizontal bars about the means) are plotted on a validation set of 100 vectors. The mean error at each point is labeled with the features retained in the neural network model. The database partitioning and confidence interval procedures discussed in Chapter I are used to estimate the average network classification errors using the best 30 of 100 realizations of neural network training. The average

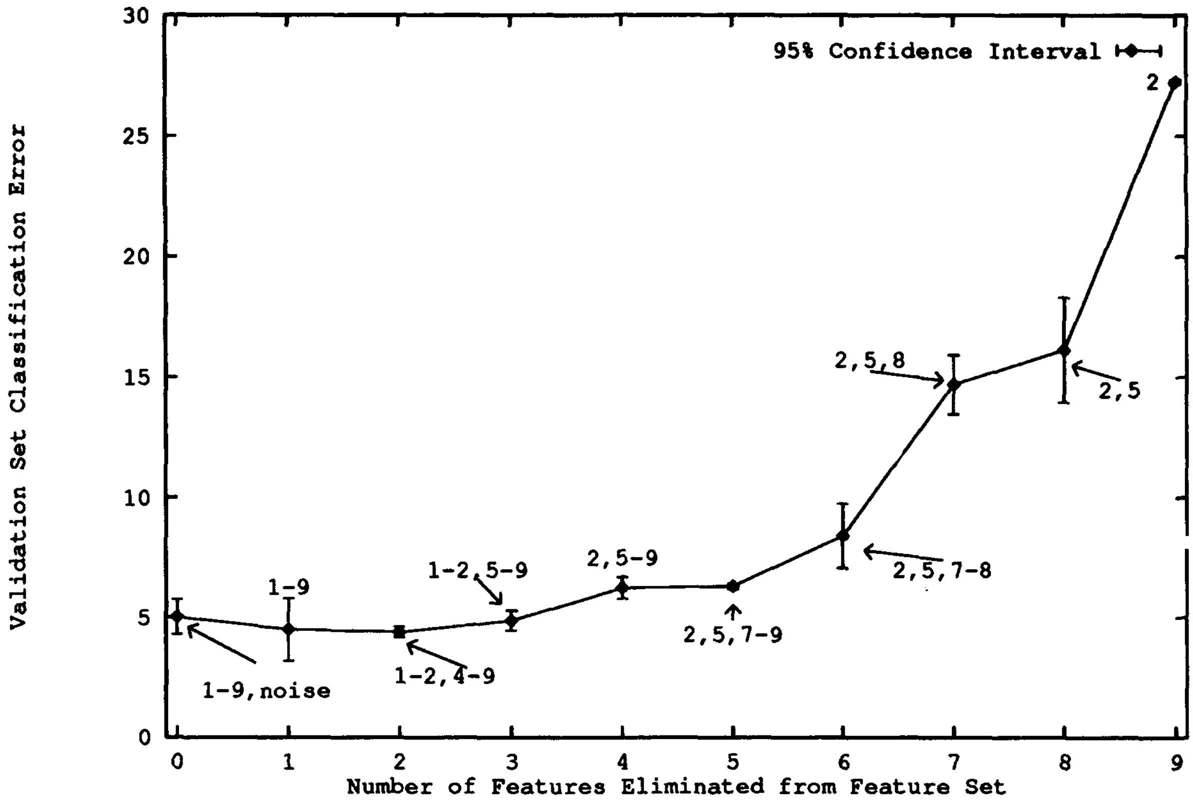


Figure 14. Average Validation Set Error on FLIR Problem as Features are Eliminated

network accuracy does not significantly change by removal of the three least salient features, including noise. In fact, the accuracy is not significantly degraded until the sixth feature is removed from the model. The explanation for this is two-fold. First, only the least salient features are removed and second, the features are intercorrelated, so a certain degree of natural redundancy may exist in this feature set.

In Figure 15, an ordered plot of the features versus the test statistics from the saliency screening procedure is shown. The relationship between 'essential' and 'nonessential' features can be seen visually. Belue and Bauer use a similar plot with features versus feature saliencies as a visual 'scree test,' to determine which features to retain [5]. For the FLIR problem, there seems to be three visual categories of features based on the test statistics. Based on using Figure 15,

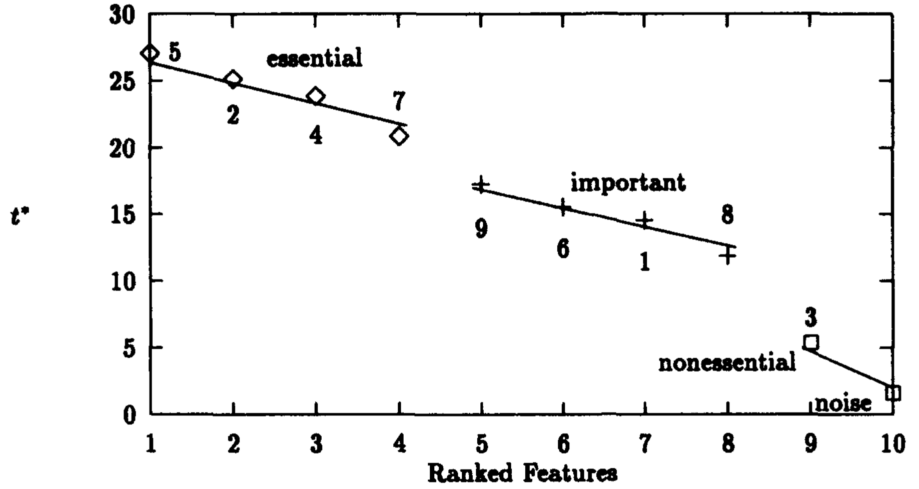


Figure 15. Scree Plot of Saliency Screening Test Statistics for FLIR Features

features 3 and noise might be categorized as 'nonessential' and removed, and the features 5, 2, 4, and 7 might be categorized as 'essential' and retained. The third category of features might be categorized as 'important' but warrant further analysis in the context of an overall feature selection process.

*4.2.7 Summary.* The saliency screening procedure is a statistical procedure for identifying nonsalient features in a feature set using a specified level of statistical confidence. A series of paired-$t$ tests are used to test for a statistical difference between the saliency of a true feature and the saliency of noise. A feature's paired-$t$ test statistic is a comprehensive measure of a feature's saliency for two reascns. First, it incorporates the relative saliency of each feature compared to a known nonsalient feature, and second, it also incorporates the variance of feature saliency.

An empirical evaluation of the saliency screening procedure indicates that conservative results are common with this procedure. That is, a nonessential feature, having little or no bearing on the classification accuracy, may be identified as salient if it is statistically different from noise. This occurs in the FLIR problem, since the third and eighth features are statistically different than noise in Table 19, but have little impact on classification accuracy as shown in Figure 14. For this reason, features with relatively low test statistics may warrant further consideration in the context of a feature selection process.

## *4.3 Weight Screening*

*4.3.1 Introduction* White describes a weight screening procedure for identifying irrelevant features [82]. The procedure uses a statistical hypothesis test which is developed using the limiting distribution of the weights $\hat{\mathbf{w}}$ [82:442]. The premise is that the vector of feature weights is not statistically different from zero for an irrelevant feature. An overview of this section follows.

First, the irrelevant input hypothesis test and the associated distributional assumptions are reviewed. Then, a relationship is shown between weight screening and weight-based saliency. Next, a weight screening methodology using the irrelevant input hypothesis test is described. The weight screening problem is then evaluated on the same problems as were previously used with the saliency screening procedure. Finally, the results are summarized along with a comparison of the two screening procedures.

*4.3.2 Background* In this section, the weight screening hypothesis test for identifying irrelevant inputs is presented. Related distributional assumptions are stated as required. The weight screening hypothesis test involves weight parameters from one realization of a trained neural network. The hypothesis test as given in White [82] follows:

$$\begin{array}{rll} \text{Null Hypothesis} & \mathbf{H}_0: & \mathbf{S}\mathbf{w}^* = \mathbf{0} \\ \text{Alternative Hypothesis} & \mathbf{H}_A: & \mathbf{S}\mathbf{w}^* \neq \mathbf{0}\ , \end{array}$$

where $\mathbf{w}^*$ is an $s$-dimensional vector of neural network optimal weight parameters and $\mathbf{S}$ is a $q$ by $s$ selection matrix picking out the $q$ elements of $\mathbf{w}^*$ which are hypothesized to be zero under $\mathbf{H}_0$.

To statistically test $\mathbf{H}_0$ for a neural network, the limiting distribution of the vector of estimated network weights $\hat{\mathbf{w}}$ must be multivariate normal as $\hat{\mathbf{w}}$ converges to $\mathbf{w}^*$ [82]. The vector of estimated weights $\hat{\mathbf{w}}$ will be multivariate normally distributed 'in the limit' if redundant feature inputs and redundant hidden units are removed [79:441]. 'In the limit' refers to weights from a network trained using an infinitely large data set. When $\hat{\mathbf{w}}$ has a multivariate normal limiting distribution, then

$$\sqrt{P}(\hat{\mathbf{w}} - \mathbf{w}^*) \sim N_s(\mathbf{0}, \mathbf{C}^*),$$

where $\mathbf{C}^*$ is the $s$-dimensional covariance matrix of $\sqrt{P}(\hat{\mathbf{w}} - \mathbf{w}^*)$ (and $\sqrt{P}\hat{\mathbf{w}}$) and $P$ is the number of data exemplars [82:441]. Also,

$$\sqrt{P}\mathbf{S}(\hat{\mathbf{w}} - \mathbf{w}^*) \sim N_q(\mathbf{0}, \mathbf{S}\mathbf{C}^*\mathbf{S}')$$

and

$$P(\hat{\mathbf{w}} - \mathbf{w}^*)'\mathbf{S}'(\mathbf{S}\mathbf{C}^*\mathbf{S}')^{-1}\mathbf{S}(\hat{\mathbf{w}} - \mathbf{w}^*) \sim \chi^2_q$$

Under the null hypothesis where $\mathbf{S}\mathbf{w}^* = \mathbf{0}$, the limiting distributions are simplified to

$$\sqrt{P}\mathbf{S}\hat{\mathbf{w}} \sim N_q(\mathbf{0}, \mathbf{S}\mathbf{C}^*\mathbf{S}')$$

and

$$P\hat{\mathbf{w}}'\mathbf{S}'(\mathbf{S}\mathbf{C}^*\mathbf{S}')^{-1}\mathbf{S}\hat{\mathbf{w}} \sim \chi^2_q \qquad (61)$$

where the $\chi^2_q$ test statistic is defined by the left hand side of Equation 61.

The analytical expression for $\mathbf{C}^*$ cannot be computed; however, a weakly consistent estimator $\hat{\mathbf{C}}$ is available [79]. The estimator denoted $\hat{\mathbf{C}}$ is defined as

$$\hat{\mathbf{C}} = \hat{\mathbf{A}}^{-1}\hat{\mathbf{B}}\hat{\mathbf{A}}^{-1} \qquad (62)$$

The $s$-dimensional matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ are

$$\hat{\mathbf{A}} = P^{-1}\sum_{p=1}^{P}\nabla^2\mathcal{E}_o^p(\mathbf{x}^p,\hat{\mathbf{w}}) \tag{63}$$

$$\hat{\mathbf{B}} = P^{-1}\sum_{p=1}^{P}\nabla\mathcal{E}_o^p(\mathbf{x}^p,\hat{\mathbf{w}})\nabla\mathcal{E}_o^p(\mathbf{x}^p,\hat{\mathbf{w}})' \tag{64}$$

where $\mathcal{E}_o^p$ is the neural network error associated with the $p$th exemplar (see Chapter I Equation 1), and the operators $\nabla$ and $\nabla^2$ denote the $s \times 1$ gradient and the $s \times s$ Hessian operators of $\mathcal{E}_o^p$. Replacing $\mathbf{C}^*$ with $\hat{\mathbf{C}}$ does not affect the limiting $\chi_q^2$ distribution; however, White warns that a much larger sample size $P$ may be required to obtain a good approximation of $\mathbf{C}^*$ [79:442-443].

The $\chi_q^2$ test statistic must be estimated using $\hat{\mathbf{C}}$. It is defined as

$$\chi_q^{2^*} = P\hat{\mathbf{w}}'\mathbf{S}'(\mathbf{S}\hat{\mathbf{C}}\mathbf{S}')^{-1}\mathbf{S}\hat{\mathbf{w}} \tag{65}$$

If $\chi_q^{2^*}$ exceeds the $1-\alpha$ percentile of $\chi_q^2$, the irrelevant input hypothesis is rejected where the probability of failing to reject $\mathbf{H_0}$ when $\mathbf{H_0}$ is false is equal to $\alpha$.

*4.3.3 Relationship between Weight Saliency and Weight Screening* In this section, a relationship is shown between the Euclidean weight-based saliency $\Upsilon_i$ proposed by Tarr (see Equation 22 in Chapter II) and the $\chi_q^2$ test statistic (on the left hand side of Equation 61) [82, 72]. Let $\mathbf{S}_i$ be the $q$ by $s$ selection matrix which selects the $q$ weights associated with feature $i$. The weight saliency $\Upsilon_i$ and the $\chi_q^2$ test statistic associated with the $i$th feature are redefined in terms of the selection matrix $\mathbf{S}_i$ as

$$\Upsilon_i = \hat{\mathbf{w}}'\mathbf{S}_i'\mathbf{S}_i\hat{\mathbf{w}},$$

$$P\hat{\mathbf{w}}'\mathbf{S}_i'(\mathbf{S}_i\mathbf{C}^*\mathbf{S}_i')^{-1}\mathbf{S}_i\hat{\mathbf{w}} \sim \chi_q^2$$

A relationship between $\Upsilon_i$ and the $\chi^2_q$ test statistic is developed as follows

$$\begin{aligned}
\Upsilon_i &= \hat{\mathbf{w}}'\mathbf{S}_i'\mathbf{S}_i\hat{\mathbf{w}} \\
&= \hat{\mathbf{w}}'\mathbf{S}_i'\mathbf{I}_q\mathbf{S}_i\hat{\mathbf{w}} \\
&= \hat{\mathbf{w}}'\mathbf{S}_i'(\mathbf{S}_i\mathbf{I}_s\mathbf{S}_i')\mathbf{S}_i\hat{\mathbf{w}} \\
&= \hat{\mathbf{w}}'\mathbf{S}_i'(\mathbf{S}_i\mathbf{I}_s\mathbf{S}_i')^{-1}\mathbf{S}_i\hat{\mathbf{w}}
\end{aligned}$$

Now, by considering the special case where $\mathbf{C}^* = \mathbf{I}_s$ and substituting $\mathbf{C}^*$ for $\mathbf{I}_s$ above gives

$$\Upsilon_i = \hat{\mathbf{w}}'\mathbf{S}_i'(\mathbf{S}_i\hat{\mathbf{C}}^*\mathbf{S}_i')^{-1}\mathbf{S}_i\hat{\mathbf{w}}$$

which is equivalent to

$$P\Upsilon_i = P\hat{\mathbf{w}}'\mathbf{S}_i'(\mathbf{S}_i\hat{\mathbf{C}}^*\mathbf{S}_i')^{-1}\mathbf{S}_i\hat{\mathbf{w}}$$

Therefore, the special case where $\mathbf{C}^* = \mathbf{I}_s$, gives the relationship

$$P\Upsilon_i \sim \chi^2_q$$

This relationship can only be shown for $\mathbf{C}^* = \mathbf{I}_s$. This fact highlights that weight saliency does not incorporate the variance covariance structure of the weights emanating from the feature of interest.

*4.3.4 Methodology.* A formalized statistical weight screening procedure for identifying irrelevant features is based on applying the Bonferroni inequality to $M$ individual hypothesis tests to achieve a predetermined 'family' significance level $\alpha$. For the weight screening procedure, the individual hypothesis test associated with the $i$th feature is given as

$$\text{Null Hypothesis} \quad \mathbf{H}_0: \quad \mathbf{S}_i\mathbf{w}^* = \mathbf{0}$$

$$\text{Alternative Hypothesis} \quad \mathbf{H}_A: \quad \mathbf{S}_i\mathbf{w}^* \neq \mathbf{0}$$

The null hypothesis is referred to by White as the 'irrelevant input hypothesis' for testing whether the $i$th feature is irrelevant [82].

The weight screening procedure can be summarized in a similar manner to the saliency screening procedure as follows:

### Weight Screening Procedure

1. Train a neural net to minimize *training-test* set error.

   The net should use a minimal network structure with no redundant feature inputs and no redundant middle nodes.

   The net should be trained to a local minimum.

2. Select 'family' significance level, $\alpha$.

3. For each of the $M$ features, do an individual hypothesis test as follows:

   (a) Compute the matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ in Equations 63 and 64.

   (b) Compute $\hat{\mathbf{C}}$ in Equation 62.

   (c) Form the appropriate selection matrix $S$ which picks out the $q$ elements of $\mathbf{w}^*$ which are hypothesized to be zero under $\mathbf{H}_0$.

   (d) Compute $\chi_q^{2*}$ test statistic in Equation 65.

   (e) Determine the Bonferroni critical value $B = \chi^2_{\frac{\alpha}{M},q}$.

   (f) Evaluate the test statistic as follows:

- If $\chi_q^{2*} \leq B$, the null hypothesis can not be rejected for feature $i$.

  Conclusion: feature $i$ is irrelevant, since the weights associated with the $i$th feature are not statistically different from 0 at the $\alpha$ 'family' significance level.

- If $\chi_q^{2*} > B$, reject the null hypothesis for feature $i$.

  Conclusion: feature $i$ is relevant, since the weights are statistically different than 0 at the $\alpha$ 'family' significance level.

4. Eliminate the irrelevant features and retrain the network using only the relevant features.

If there are any questions concerning the salient features, the procedure can be performed again on the reduced feature set to confirm the remaining features are all still salient.

There are two potential problems with practical implementation of the weight screening procedure:

1. Ill-conditioned matrices due to redundant inputs and/or middle nodes.
2. Intractable matrix inversion due to a very large network structure.

When there are redundant inputs or redundant middle nodes, the weights $\hat{\mathbf{w}}$ are no longer multivariate normally distributed; therefore, the $\chi_q^{2*}$ test statistic is no longer distributed as a $\chi_q^2$ under the null hypothesis. If ill-conditioning is present or suspected, one should proceed with caution. Ideally, all redundancies should be eliminated in order to perform the irrelevant input hypothesis test with the $\chi_q^{2*}$ test statistic. The second potential problem can not always be avoided or eliminated. The computation of $\chi_q^{2*}$ may become intractable even when the network is trained and redundancies in the features and middle nodes have been eliminated. The intractability is primarily due to the inversion of the s-dimensional matrix $\hat{\mathbf{A}}$ which is required for estimating the covariance matrix $\mathbf{C}^*$.

The practical limitations on the problem size can be mitigated by making simplifying assumptions about the covariance structure of the weights. These assumptions will in turn simplify the process of inverting the Hessian matrix $\hat{\mathbf{A}}$. However, improving the practical implementation when there are size limitations only makes sense if the procedure is useful for consistently identifying irrelevant features. The performance of the weight screening procedure is evaluated in the next section.

*4.3.5 Analysis* In this section, an empirical evaluation oi the weight screening procedure is presented using the XOR, API, and FLIR problems. For all three problems, results are presented for the same 30 runs which were used with the saliency screening in Section 2. All of the same training sets, training parameters, and final weight estimates are used.

Recall the XOR problem which was augmented with five additional noise features (all treated as candidate features). The weight screening hypothesis test results for the XOR problem are summarized in Table 20. The number of times the null hypothesis was rejected in 30 neural network runs is reported in the first column of Table 20. For example, the null hypothesis was rejected for feature $x$ in 25 out of 30 runs. This means that in five of the runs, the vector of weights connected to feature $x$ were not statistically different than 0. Depending on which run is used, the 'true' features $x$ and $y$, may or may not be statistically different than noise. In fact, for one of the network runs the test statistic from one of the noise features was larger than the test statistic from one of the 'true' features. The average $\chi^{2^*}$ test statistics over 30 runs are reported in the second column of Table 20. While the results shown in column one indicate what might occur on any given run, the results shown in column two indicate what can be gleaned if the information from all 30 runs is combined. On this problem, the average test statistics indicate the features $x$ and $y$ are relevant, and the noise features are irrelevant.

The second problem used is the API problem which is augmented with two additional noise features (both treated as candidate features). The weight screening hypothesis test results are summarized in Table 21. The number of times the null hypothesis was rejected in 30 neural network runs is reported in the first column of Table 21. On any individual run, the weight screening results

Table 20. Weight Screening Results on XOR Problem for 30 Runs

| Feature | $H_0$ Rejections out of 30 | Average $\chi^{2*}$ |
|---|---|---|
| $x$ | 25 | 139.57 |
| $y$ | 22 | 145.78 |
| noise 1 | 3 | 4.74 |
| noise 2 | 0 | 3.70 |
| noise 3 | 1 | 4.31 |
| noise 4 | 0 | 3.89 |
| noise 5 | 3 | 5.32 |
| Individual Significance Level .005<br>Critical Value $\chi^2_q = 14.86$, where $q = 4$ | | |

may or may not be consistent with the saliency screening results. That is, the noise features and $M_S$ may or may not be irrelevant and the other 'salient' features may or may not be identified as irrelevant. The average $\chi^{2*}$ test statistics over 30 runs are reported in the second column of Table 21. On this problem, the average test statistics indicate that only the noise features are irrelevant. The feature $M_S$ probably warrants further investigation, since its test statistic is fairly close to the critical value.

The third problem used is the FLIR problem which was augmented with two additional noise features (both treated as candidate features). The weight screening hypothesis test results are summarized in Table 22. The number of times the null hypothesis was rejected over 30 neural network runs is reported in the first column of Table 22. These results indicate that none of the features are consistently relevant. In fact, on several of the 30 simulations none of the features have weights which are statistically relevant (i.e. weights that are statistically different than 0). The average $\chi^{2*}$ test statistics over 30 runs are reported in the second column of Table 22. For this

Table 21. Weight Screening Results on API Problem for 30 Runs

| Feature | $H_0$ Rejections out of 30 | Average $\chi^{2*}$ |
|---|---|---|
| *PLY* | 29 | 126.3 |
| $V_S$ | 25 | 51.8 |
| $M_S$ | 14 | 20.5 |
| *SECT* | 29 | 106.67 |
| noise 1 | 6 | 11.21 |
| noise 2 | 11 | 15.61 |
| Individual Significance Level .005 | | |
| Critical Value $\chi^2_q = 16.75$, where $q = 5$ | | |

problem, the average test statistics indicate that only three features are relevant. The fifth feature which has the largest average $\chi^2$ test statistic also has the largest saliency screening test statistic in Table 19. However, feature two which has the second largest saliency test statistic in Table 19 was identified as relevant only seven times out of the thirty runs using the weight screening procedure.

On all of these problems, the weight screening methodology did not provide consistent reliable results for the 30 runs evaluated. For the first two problems, noise was not consistently identified as irrelevant. Using the weight screening procedure resulted in identifying some or even all of the true variables as irrelevant on some individual runs.

Although the weight screening procedure is meant to be used with just a single neural network run, the average test statistics over 30 runs has more potential as a screening tool. For the XOR and API problems, the average test statistics indicate that all of the features are relevant except for noise. However, when using the average test statistics for the FLIR problem, none of the features are relevant except feature five.

Table 22. Weight Screening Results on FLIR Problem for 30 Runs

| Feature | $H_0$ Rejections out of 30 | Average $\chi^{2*}$ |
|---|---|---|
| 1 | 4 | 10.6 |
| 2 | 7 | 10.3 |
| 3 | 1 | 2.0 |
| 4 | 12 | 14.8 |
| 5 | 23 | 39.8 |
| 6 | 5 | 7.2 |
| 7 | 11 | 15.3 |
| 8 | 5 | 8.4 |
| 9 | 0 | 2.5 |
| noise 1 | 1 | 3.2 |
| noise 2 | 1 | 3.1 |
| Individual Significance Level .005<br>Critical Value $\chi^2 = 14.86$, where $q = 4$ | | |

*4.3.6 Summary* The weight screening procedure, while a single run procedure, is not robust. It does not provide consistent reliable results for a single run This indicates the procedure is extremely sensitive to less than ideal conditions where the weights may not be multivariate normally distributed in the limit or may not be from a perfectly trained network. Further, the weight screening procedure is computationally impractical for reasonable sized networks due to the required Hessian matrix inversion.

In the identification of noise, the average test statistics over several runs are more useful than the test statistics from a single run. However, in the case of the FLIR problem, only one feature is identified as relevant. As with the saliency screening procedure, a visual comparison of the average test statistics might be useful to categorize features as 'relevant' and 'irrelevant' for some problems.

### *4.4 Summary*

Two statistically based feature screening procedures are discussed in this chapter. The two statistically based methods for the identifying nonsalient/irrelevant features are the saliency screening procedure and the weight screening procedure. In the saliency screening procedure, a feature's mean saliency is statistically compared to the mean saliency of a known noise feature. In the weight screening procedure, the weights emanating from a feature are tested to see if they are statistically different than 0. With either of these statistical screening procedures, features can be ranked and categorized as 'essential' and 'nonessential' based on their test statistics.

It is desirable for a feature screening procedure to be able to consistently identify noise or irrelevant features as different from other features. An evaluation of both procedures demonstrates that the saliency screening procedure is more robust than the weight screening procedure, since it consistently identifies *noise-like* features as being nonsalient/irrelevant whereas the the weight screening procedure does not. Additionally, disadvantages are associated with the weight screening procedure which are not associated with the saliency screening procedure. A summary of the disadvantages follows.

- The procedure requires the weight parameters to have a multivariate normal limiting distribution, which is often violated in the practical implementation of neural networks.
- The procedure relies on a weakly consistent estimate of the limiting covariance matrix of the weights.
- The procedure for estimating the limiting covariance matrix of the weights requires the second derivative information from the Hessian matrix.
- The procedure has practical limitations on the problem size, since the Hessian matrix must be inverted.
- The irrelevant noise features are not consistently identified with this procedure.

The statistical saliency screening procedure presented in this chapter is useful for identifying and eliminating noisy features from a set of candidate features up front prior to using a feature selection procedure for investigating reduced feature subsets. In the next chapter, two novel neural network selection algorithms are presented for investigating reduced feature subsets and appropriate neural network architecture.

# *V. Selection Algorithms for Neural Networks*

## *5.1 Introduction*

In this chapter, a statistical model selection perspective is adopted for neural networks. Algorithms are developed for automating the process of model selection for feedforward neural networks. Both algorithms are developed based on nonlinear regression model selection criteria within a backwards sequential procedure. The first is an initial architecture selection algorithm for determining an appropriate number of middle nodes. The second is a feature selection algorithm for reducing feature sets which may be either larger than necessary, or larger than desired. An overview of the chapter follows.

Univariate response nonlinear regression, model selection criteria, and practical considerations for neural network model selection are reviewed in Section 2. In Section 3, a backwards sequential selection algorithm, and neural network algorithms for architecture and feature selection are presented. The results in this chapter are summarized in Section 4.

## *5.2 Univariate Nonlinear Regression and Model Selection*

*5.2.1 Introduction.* "Backpropagation and nonlinear regression can be viewed as alternative statistical approaches to solving the least squares problem" [80:85]. The major difference between instantaneous backpropagation least squares nonlinear regression is that in least squares nonlinear regression, the weight parameters are determined with techniques which use "batched" updates rather than "instantaneous" updates which use one data point at a time. In this section, a single output neural network trained with backpropagation is formalized by adopting the framework of a univariate nonlinear least squares regression model [80:85-89]. Using this framework, nonlinear regression statistical model selection criteria are defined. Practical considerations for neural network model selection are discussed.

*5.2.2 The Neural Network Nonlinear Regression Model.* Consider a single output neural network as a univariate response nonlinear regression model.

$$\mathbf{d} = \mathbf{z}(\mathbf{X}, \mathbf{w}^*) + \boldsymbol{\epsilon} \tag{66}$$

where **d** is the $P \times 1$ vector of true 'desired' network outputs; $\mathbf{z}(\mathbf{X}, \mathbf{w})$ can be interpreted as $\mathbf{E}(\mathbf{d}|\mathbf{X})$ the $P \times 1$ vector of neural network responses conditioned on the $P \times M$ matrix of feature input variables **X**; $\mathbf{w}^*$ is the $s \times 1$ vector of unknown optimal weight parameters; and $\boldsymbol{\epsilon}$ is the $P \times 1$ vector of neural network errors. The least squares estimator of $\mathbf{w}^*$ is the $s \times 1$ vector $\hat{\mathbf{w}}$ that minimizes the neural network sums of squared errors (SSE) with respect to $\mathbf{w}$.

$$\hat{\mathbf{w}} = \arg\min \{\mathrm{SSE}(\mathbf{w})\},$$

where

$$\mathrm{SSE}(\mathbf{w}) = [\mathbf{d} - \mathbf{z}(\mathbf{X}, \mathbf{w})]'[\mathbf{d} - \mathbf{z}(\mathbf{X}, \mathbf{w})] \tag{67}$$

An estimate of the variance of the errors $\boldsymbol{\epsilon}$ corresponding to the least squares estimator $\hat{\mathbf{w}}$ is

$$\hat{\sigma}^2 = \frac{\mathrm{SSE}(\hat{\mathbf{w}})}{P - s}$$

When there are no isolated flats in the parameter space caused by redundant inputs or middle nodes, the unknown optimal weight vector $\mathbf{w}^*$ is said to be locally unique [80:106]. The neural network model is said to be locally identifiable, that is $\mathbf{w}^*$ is a unique local solution, and the estimated weight vector $\hat{\mathbf{w}}$ has a multivariate normal distribution [80:106-7].

A discussion of conditional probability laws as they pertain to the interpretation of neural networks can be found in White [80:95]. In the discussion, White recognizes the special case of the binary dependent variable **d** associated with any two-way classification problem [80:95]. Minimizing the squared errors with backpropagation produces a minimum mean-squared error estimator $\hat{\mathbf{w}}$

and a minimum mean-squared error approximation to $\mathbf{E}(\mathbf{d}|\mathbf{X})$ [80:98]. By the definition of $\boldsymbol{\epsilon} = \mathbf{d} - \mathbf{z}(\mathbf{X}, \mathbf{w}^*)$ and the properties of conditional expectation, the expectation of $\boldsymbol{\epsilon}$ conditioned on $X$ is zero [82:430]. With the added assumption of normally distributed errors, least squares nonlinear regression is equivalent to the method of maximum likelihood nonlinear regression [80:259].

*5.2.3 Three Selection Criteria.* There are three model selection criteria for correctly specified nonlinear regression models which are associated with normally distributed errors. They are the Wald, Lagrange multiplier, and likelihood ratio tests. Taken in the context of neural network model selection, all of these tests can be defined with the same general hypothesis test:

$$\begin{aligned} \text{Null Hypothesis} \quad & \mathbf{H}_0: \quad \mathbf{S}\mathbf{w}^* = \mathbf{0} \\ \text{Alternative Hypothesis} \quad & \mathbf{H}_A: \quad \mathbf{S}\mathbf{w}^* \neq \mathbf{0}\ , \end{aligned}$$

where $\mathbf{w}^*$ is an $s \times 1$ vector of neural network optimal weight parameters and $\mathbf{S}$ is a $q \times s$ selection matrix picking out the $q$ elements of $\mathbf{w}^*$ which are hypothesized to be zero under $\mathbf{H}_0$.

All three tests have an associated model selection criterion, sometimes referred to as a test statistic. These are denoted as: $W$ for the Wald test statistic, $R_1$ or $R_2$ for the Lagrange multiplier test statistics, and $L$ for the likelihood ratio test statistic. In this section, these test statistics are defined in neural network terminology.

The Wald test statistic is adapted from Gallant [20:48], and White [80:109].

$$W = \frac{\hat{\mathbf{w}}'\mathbf{S}'(\mathbf{S}(\hat{\mathbf{F}}'\hat{\mathbf{F}})^{-1}\mathbf{S}')^{-1}\mathbf{S}\hat{\mathbf{w}}\,/\,q}{\mathrm{SSE}\,/\,P - s} \tag{68}$$

where $\hat{\mathbf{F}} = \mathbf{F}(\hat{\mathbf{w}})$ and where $F$ is defined

$$\mathbf{F}(\hat{\mathbf{w}}) = \frac{\partial}{\partial \mathbf{w}'}\mathbf{z}(\mathbf{X}, \hat{\mathbf{w}}) \tag{69}$$

and SSE is the sum of squared-errors defined in Equation 67 which is evaluated for the full model. Under the null hypothesis, $W$ in Equation 68 is approximately $F$ distributed with $q$ numerator and $P - s$ denominator degrees of freedom [20:48].

For the Lagrange multiplier test statistics, the SSE is minimized for the full model subject to the null hypothesis. This is a constrained minimization where the resulting estimator is denoted $\tilde{\mathbf{w}}$. The unconstrained minimization of SSE results in the estimator $\hat{\mathbf{w}}$. Using the constrained estimator $\tilde{\mathbf{w}}$ as a starting value, one 'unconstrained' Gauss-Newton step is taken towards $\hat{\mathbf{w}}$. In neural network terms, a Gauss-Newton step is equivalent to one epoch of instantaneous updates or one batch update. The Gauss Newton step, denoted $\tilde{\mathbf{D}}$ is defined in Gallant [20:85] as

$$\tilde{\mathbf{D}} = \left(\tilde{\mathbf{F}}'\tilde{\mathbf{F}}\right)^{-1}\tilde{\mathbf{F}}'\left[\mathbf{d} - \mathbf{z}(\mathbf{X}, \tilde{\mathbf{w}})\right],$$

where $\tilde{\mathbf{F}} = \mathbf{F}(\tilde{\mathbf{w}})$, similar to Equation 69. If $\mathbf{H}_0$ is true, then $\tilde{\mathbf{D}}$ will be small. Conversely, if $\mathbf{H}_0$ is false, then $\tilde{\mathbf{D}}$ will be large.

There are two versions of the Lagrange multiplier test statistics, $R_1$ and $R_2$. Each is based on a measure of $\tilde{\mathbf{D}}$. The definitions of $R_1$ and $R_2$ are adapted from Gallant [20:86].

$$\begin{aligned} R_1 &= \frac{\tilde{\mathbf{D}}'\left(\tilde{\mathbf{F}}'\tilde{\mathbf{F}}\right)\tilde{\mathbf{D}}/q}{\mathrm{SSE}(\hat{\mathbf{w}})/(P-s)} \\ R_2 &= \frac{n\tilde{\mathbf{D}}'\left(\tilde{\mathbf{F}}'\tilde{\mathbf{F}}\right)\tilde{\mathbf{D}}}{\mathrm{SSE}(\tilde{\mathbf{w}})}, \end{aligned}$$

where SSE($\tilde{\mathbf{w}}$) represents the constrained minimum sum of squared errors associated with fitting the full model subject to $\mathbf{H}_0$, and SSE($\hat{\mathbf{w}}$) represents the minimum sum of squared errors after taking one unconstrained Gauss Newton step. When using the first version of the Lagrange multiplier test statistic, the null hypothesis is rejected when $R_1$ exceeds the $\alpha \times 100$ critical point of the $F$-distribution, denoted $F_{1-\alpha;q,P-s}$. When using the second version of the Lagrange multiplier test statistic, the null hypothesis is rejected when $R_2$ exceeds $d_\alpha$, where

$$d_\alpha = \frac{nF_\alpha}{(P-s)/(q) + F_\alpha}$$

The likelihood ratio test involves fitting two nonlinear regression models. The full model is fit under the alternative hypothesis, and the reduced model is fit under the null hypothesis. The full model has $P - s$ degrees of freedom, and the reduced model has $P - (s - q)$ degrees of freedom, where $q$ is the number of weight parameters constrained to be sero in the reduced model. The associated likelihood ratio test adapted from Gallant [20:56] is:

$$L = \frac{(\mathrm{SSE}_R - \mathrm{SSE}_F)/q}{(\mathrm{SSE}_F)/(P-s)}, \tag{70}$$

where $\mathrm{SSE}_F$ and $\mathrm{SSE}_R$ are the full and reduced models' sums of squared-errors. Under $\mathbf{H}_0$, $L$ is $F$-distributed with $q$ numerator and $P - s$ denominator degrees of freedom. Therefore, whenever $L$ exceeds the $\alpha \times 100$ critical point of the $F$-distribution, the null hypothesis is rejected.

*5.2.4 Two Specification Robust Selection Criteria.* The neural network model is generally an incorrectly specified probability model and normally distributed errors can not be assumed. White "explores the ramifications of maximizing a general likelihood function that is not derived from a correctly specified probability model - precisely the situation under which network learning generally occurs" [80:259-288].

For situations where the errors are non-normal or the model is not correctly specified, White describes specification robust versions of the Wald and Lagrange multiplier tests which are asymptotically distributed as $\chi^2_q$ [80:263-267]. The robust version of the Wald test statistic for neural network model selection is identical to the weight screening $\chi^2_q$ test statistic presented in Chapter IV [80:109,266]. The robust versions of both of these test statistics are given in White [80:266-267]. Under misspecification, the likelihood ratio statistic is, generally, not equivalent to the Wald or

Lagrange multiplier test statistic, nor is it asymptotically distributed as $\chi_q^2$ (in the $\chi_q^2$ version of the test statistic considered by Gallant) [80:267], [20:591].

The primary difference between the specification robust and non-specification robust versions of the Wald and Lagrange multiplier test statistics is that the weight parameter covariance matrix is no longer approximated by $\hat{\sigma}^2(\hat{\mathbf{F}}'\hat{\mathbf{F}})^{-1}$. Instead, the matrix is approximated by $P^{-\frac{1}{2}}\mathbf{C}^*$. An analytical expression for $\mathbf{C}^*$ is not available, but an estimator $\hat{\mathbf{C}}$ exists which is weakly consistent. The estimator is defined as

$$\hat{\mathbf{C}} = \hat{\mathbf{A}}^{-1}\hat{\mathbf{B}}\hat{\mathbf{A}}^{-1} \tag{71}$$

The matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ are defined

$$\begin{aligned} \hat{\mathbf{A}} &= P^{-1}\sum_{t=1}^{P} \nabla^2\mathcal{E}(\mathbf{x}_t, \hat{\mathbf{w}}) \\ \hat{\mathbf{B}} &= P^{-1}\sum_{t=1}^{P} \nabla\mathcal{E}(\mathbf{x}_t, \hat{\mathbf{w}})\nabla\mathcal{E}(\mathbf{x}_t, \hat{\mathbf{w}})' \end{aligned}$$

where $P$ is the sample size; $\mathcal{E}$ is the neural network error used for training; the operator $\nabla$ is the $s \times 1$ gradient of $\mathcal{E}$ with respect to $\hat{\mathbf{w}}$, and the operator $\nabla^2$ is the $s \times s$ Hessian operator of $\mathcal{E}$ with respect to $\hat{\mathbf{w}}$, where $\mathbf{x}_t$ is the $t$-th input exemplar. Replacing $\mathbf{C}^*$ with $\hat{\mathbf{C}}$ does not affect the limiting $\chi_q^2$ distribution; however, White warns that a much larger sample sise may be required to obtain a good approximation of $\mathbf{C}^*$ [79:442-443].

Some disadvantages associated with the specification robust Wald statistic are documented for its application to neural network feature screening in Chapter IV. They are reviewed here for completeness:

- The procedure requires the weight parameters to have a multivariate normal limiting distribution. Due to redundant inputs and middle nodes, this is often violated in the practical implementation of neural networks.

- The procedure relies on a weakly consistent estimate of the limiting covariance matrix of the weights.
- The procedure for estimating the limiting covariance matrix of the weights requires second derivative information from the Hessian matrix.
- The procedure has practical limitations on the problem size since the Hessian matrix must be inverted.
- Irrelevant noise features are not consistently identified with this procedure (see results shown in Chapter IV, Section 3.5).

For the specification robust version of the Lagrange multiplier test, these disadvantages remain. However, no decisive statement can be made regarding the last disadvantage, since use of the Lagrange multiplier test for neural network model selection has not been documented.

*5.2.5 Practical Considerations for Neural Network Selection.* For practical implementation by a neural network practitioner, the specification robust versions of model selection criteria are associated with computationally burdensome test statistics and restrictive assumptions which are often compromised in a model selection scenario.

The required estimation of the covariance matrix of the weights involves the formation and subsequent inversion of the (possibly ill-conditioned) Hessian matrix. This is computationally burdensome, and puts a practical limitation on the size of the network to be analyzed

When there are no redundancies in either the network structure or feature inputs the required normality assumption is satisfied [80:105]. White assumes control can be exercised over the limiting multivariate distribution of the weight parameters and that the neural network practitioner has expended the effort necessary to remove feature and middle node redundancies [80:106-108]. However, redundancies are not always easy to detect or remedy. This fact has been well documented for the phenomena of multicollinearity in linear regression [46]. Further, the specification robust tests are not appropriate for identifying unnecessary redundant parameters [80:109,145].

Consider a neural network feature selection scenario where irrelevant noise features have been identified and removed through the feature screening procedures defined in Chapter IV. In this feature selection scenario, any candidate features which should be eliminated are probably redundant. This is precisely the scenario where the specification robust test statistics have problems, since the asymptotic normality assumptions do not hold in the presence of redundancies.

Since the advantage of specification robust test statistics is generally compromised in a feature selection scenario, the "non-specification robust" test statistics are considered further. Of the three "non-specification robust" test statistics discussed in Section 2, the likelihood ratio test is the most appealing for practical use. It is appealing for the following reasons:

- The likelihood ratio test is not affected by parameter effects curvatures which are associated with the linearized approximation of $\mathbf{z}(\mathbf{X}, \mathbf{w})$ used for the Wald statistic [66:200-220].
- The likelihood ratio test is more powerful than either version of the Lagrange ratio test statistics [20:89].
- The likelihood ratio test statistic is the only test statistic which does not require estimating and subsequently inverting the covariance matrix of the weight parameters.
- The likelihood test statistic is based on comparing minimum sums of squared-errors which is the basis for parameter estimation in standard backpropagation training.
- The likelihood ratio test statistic is prominently used in linear regression model selection algorithms.

### *5.3 Neural Network Selection Algorithms*

*5.3.1 Introduction.* Two neural network model selection algorithms are proposed in this section: the initial architecture selection algorithm and the feature selection algorithm. Both algorithms are developed by adopting a statistical model building perspective using a backwards sequential procedure. For completeness, the backwards sequential selection algorithm will be reviewed before it is used with the model selection algorithms. The backwards sequential selection algorithm is proposed in this research because potentially better features may be selected when correlated features are present [43].

Due to the practical considerations discussed in Section 2, the likelihood ratio test statistic is used in this research. Since neural network models do not generally have normally distributed errors, the likelihood ratio test statistic $L$ in Equation 70 will not be exactly $F$-distributed. The assumption is that $L$ will be approximately $F$-distributed, and that the relative magnitudes of the likelihood test statistics are more palatable if one applies the $F$ distribution at some appropriately conservative significance level. The assumptions about the approximate distribution of $L$ give the proposed selection algorithms a heuristic flavor.

Neural networks trained with backpropagation can get stuck at local minima or saddle points, or they can diverge [80:143]. White recommends several neural networks be trained, and the network which has converged to the smallest error be used [80:143]. To be more specific, several nets should be trained using a different random initialization of the weight parameters and the order of presentation. When multiple neural networks are used, the minimum error network will usually yield consistent parameter estimates for some local minima, but there is still no guarantee of being close to a global minima [80:143]. Using multiple neural networks helps to minimize the probability of accepting a network which has not converged to a *good* local minima. For this reason, both of the neural network selection algorithms proposed in this section use multiple neural networks. From among the neural networks, a best network is selected based on the minimum total $SSE$ of the training and test sets. The number of neural networks one can afford i any given situation is usually limited by the time and computing resources available. A reasonable number of neural networks, five or ten, is usually sufficient for finding a *good* local minimum.

*5.3.2 Backwards Sequential Selection* The likelihood ratio test statistic can be used within a sequential search algorithm for model selection. The backwards sequential selection algorithm is proposed in this research because potentially better features may be selected when correlated features are present [43]. For completeness, a backwards sequential feature selection algorithm is outlined.

### Backward Sequential Selection

1. Set $k = M$, where $M$ is the total number of candidate features.

2. Choose a significance level $\alpha$ for feature elimination.

3. Estimate the full model with $k$ candidate features. Associated with this full model is $\text{SSE}_F$ for univariate models and $T_F$ for multivariate response models.

4. Set $p = k - 1$, where $p$ is the number of features in the reduced model.

5. Estimate all models with $p$ features which do not include any previously eliminated features. Associated with each of the $k$ reduced models is $\text{SSE}_R$ for univariate models and $T_R$ for multivariate response models.

6. Compute the likelihood ratio test statistic using Equation 70 for each of the $k$ models of $p$ features.

7. Select as a candidate feature for elimination, the feature which when removed produces the model with the lowest likelihood ratio test statistic $L$.

8. If $L < F_\alpha$ (for appropriate numerator and denominator degrees of freedom), eliminate the candidate feature, set $k = k - 1$ and go to Step 3.
   Otherwise, go to Step 9 and do not eliminate the candidate feature.

9. Stop.

*5.3.3 Architecture Selection Algorithm.* In practice, the neural network practitioner does architecture selection through a trial and error process using an informal set of "pilot runs." The pilot runs consist of a series of neural networks trained with a varying number of middle nodes. Quite often, the practitioner selects the architecture associated with the smallest network structure which can be used for accurate prediction. However, no specific criteria are universally accepted for automatically identifying a good initial network structure.

In this section, a formal architecture selection algorithm is proposed for a univariate response neural network. The algorithm uses the concepts of statistical model selection described in Section 2 to evaluate an organized series of pilot runs. Reduced neural network architectures are systematically investigated. At each step of the algorithm, a likelihood ratio test statistic is evaluated.

Figure 16 depicts the neural network architecture algorithm for a univariate response neural network. A description of the algorithm follows.

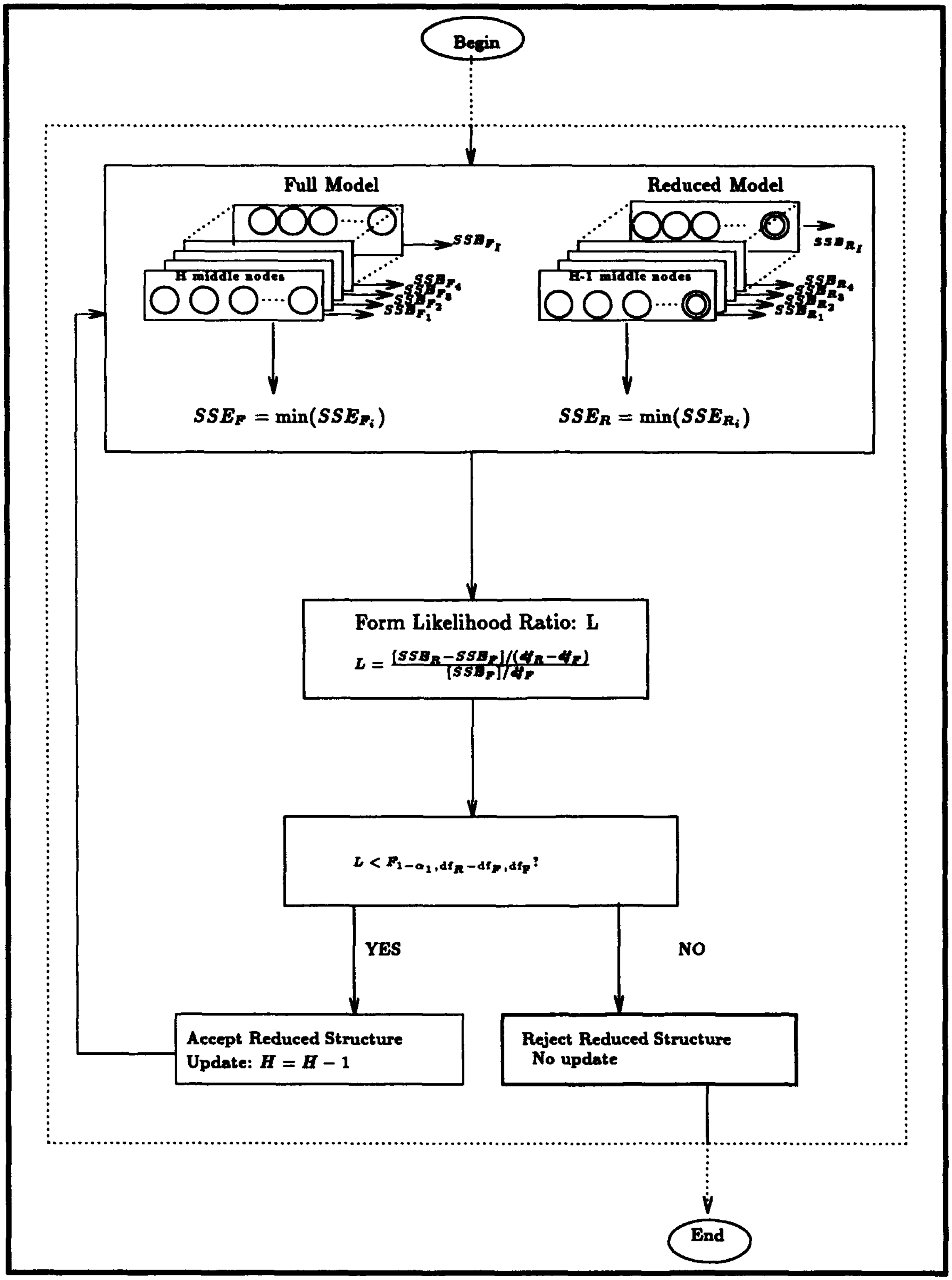
### Neural Network Architecture Selection Algorithm

1. Initialize and define parameters.

   - Number of middle nodes: $H$.
   - Significance level for statistical testing of reduced structure models: $\alpha$.
   - Number of neural network runs: $I$.
   - Number of network features: $M$.
   - Number of training set exemplars: $P_{tr}$.
   - Number of training-test set exemplars: $P_{ts}$
   - Total number of exemplars: $P = P_{tr} + P_{ts}$
   - Number of network weight parameters: $s$.

     $s = (M+1)H + (H+1)$

Figure 16. Neural Network Architecture Selection Algorithm



2. **Train $I$ realizations of the full neural network model with $H$ middle nodes.**

    - **Compute the SSE for $P$ exemplars using Equation 67.**

      **Define $\mathrm{SSE}_{F_i}$ = SSE for the $i$th neural network realization.**

      **Define $\mathrm{SSE}_F = \min\{\mathrm{SSE}_{F_1}, \cdots, \mathrm{SSE}_{F_I}\}$**

3. **Train $I$ realizations of the reduced neural network model with $H - 1$ middle nodes.**

    - **Compute the SSE for $P$ exemplars using Equation 67.**
    - **Define $\mathrm{SSE}_{R_i}$ = SSE for the $i$th neural network realization.**
    - **Define $\mathrm{SSE}_R = \min\{\mathrm{SSE}_{R_1}, \cdots, \mathrm{SSE}_{R_I}\}$.**

4. **Compute degrees of freedom.**

    - **Full model: $\mathrm{df}_F = P - s$.**
    - **Reduced model: $\mathrm{df}_R = P - [s - (M + 1)]$.**

5. **Compute the Likelihood Ratio Test Statistic: $L$.**

    - $L = \frac{[\mathrm{SSE}_R - \mathrm{SSE}_F]/(\mathrm{df}_R - \mathrm{df}_F)}{\mathrm{SSE}_F/\mathrm{df}_F}$.

6. **Test the null hypothesis that the reduced model is equivalent to the full model using $L$ defined in Equation 70.**

    - **If $L \leq F_{\alpha,\mathrm{df}_R - \mathrm{df}_F,\mathrm{df}_F}$, the reduced model can not be rejected.**
        - **Accept the reduced structure model.**
        - $H = H - 1$
        - $s = s - [M + 2]$
        - **Go to Step 2.**
    - **If $L > F_{\alpha,\mathrm{df}_R - \mathrm{df}_F,\mathrm{df}_F}$, then reject the reduced model.**

- Go to Step 7.

7. Stop, an appropriate network structure has been determined with $H$ middle nodes.

In Step 1, parameters are initialized and defined. A reasonable number of neural networks is usually five or ten. When trying to find *good* or improved local minima, there is a point of diminishing returns for the computational cost of running additional networks.

In Steps 2 and 3, each training realization involves a different training and training-test set which is randomly selected from among the total pool of exemplars. Randomly selected training and training-test sets means that each neural network run is trained using a unique partitioning of the data. Normal proportions of splitting the training and training-test sets is a $\frac{2}{3}$, $\frac{1}{3}$ split [76:30]. With smaller data sets, it is common to use a larger portion of the data in the training set [76:30].

To fairly compare results, the $SSE$ is computed in Steps 2 and 3 using all $P$ exemplars, so the full and reduced models are compared using the same data set. If there is sufficient data, a validation set $SSE$ can be used to provide the most unbiased comparison between models.

A finite number of full and reduced model networks are trained. From this, the networks corresponding to the best full and reduced model local minima are used to form the likelihood ratio in Step 5. In this scenario, where only a finite number of networks are trained, it is possible that the reduced model may produce a lower total $SSE$ than the full model. If the likelihood test statistic is negative in Step 6, the reduced model should be accepted, since the best reduced model from $I$ runs is better than the best full model from $I$ runs.

The architecture selection algorithm is generally initialized with more than enough middle nodes (i.e. some middle nodes are redundant). Redundant middle nodes in a neural network's structure cause flats in the parameter space, which may cause premature convergence to *undesirable* local minima. Multiple runs are used to avoid using neural networks which have not converged to *good* local minimum.

Some prior knowledge about the complexity of the problem at hand is useful for setting a reasonable number of initial middle nodes for the architecture selection algorithm. The number of middle nodes needs to be large enough to ensure there is sufficient complexity for accurate prediction on a test data set. However, an excessive number of middle nodes will unnecessarily increase the computational cost, and the possibility of accepting a model with more middle nodes than necessary.

One way to determine the maximum number of middle nodes for a given situation is to examine the degrees of freedom for the full model $\mathrm{df}_F$ in Step 4. Since $\mathrm{df}_F$ must be greater than zero, an upper bound for $H$ can easily be derived as

$$H << \frac{P-1}{M+1} \tag{72}$$

where $P$ is the number of training exemplars, and $M$ is the number of feature inputs not including the bias term. This upper-bound is much greater than the more analytically-based upperbound developed by Cover [10].

Cover's rule is based on the separating capacities of families of nonlinear decision surfaces [10]. Cover shows that a family of surfaces having $s$ degrees of freedom has a natural separating capacity for $2s$ training exemplars [10]. Therefore, unless the number of training exemplars is greater than the separating capacity of $2s$, there is a large probability of ambiguous generalization [10]. Cover's theorem translates into a more restrictive number of middle nodes:

$$H < \frac{.5P-1}{M+1} \tag{73}$$

It is possible that when the number of candidate features is large and the number of exemplars is small, Cover's rule may indicate fewer middle nodes than are required for the complexity of the problem at hand (a situation where feature reduction is necessary). In this case, one should proceed with caution since the neural network's ability to generalise to unknown data can be easily compromised in this situation.

There are situations where the required complexity is unclear, and the practitioner is not constrained by a small data set. One option to the practitioner is to use an excessively large number of middle nodes within the limits defined by Equation 73. However, this increases the possibility of accepting a model with more middle nodes than necessary. Initial experimentation is one method to use for determining an appropriate initialization for the middle nodes.

When computing resources are available, the architecture selection algorithm is best utilized by performing more than one initialization of $H$. Performing multiple architecture selection runs provides the practitioner with additional insight and may reduce the risk of potentially accepting a larger than necessary network structure. The downside is that this approach can also complicate the goal of automatic architecture selection, since multiple architecture selection runs may produce multiple network structures. In general, when choosing from a variety of network structures which all have acceptable prediction accuracies, the smallest network structure is preferred.

*5.3.4 Application of the Architecture Selection Algorithm* The architecture selection algorithm was applied to the API problem used in Chapter IV to determine whether a reduction in the network architecture could be justified. Belue and Bauer found eight middle nodes were sufficient for good neural network performance [6]. Using this prior knowledge, the number of middle nodes was initialized at eight. The networks are trained for a minimum of 500 epochs and improvement in the test set $SSE$ is monitored every 50 epochs to justify continuation of training. Once the test set $SSE$ no longer improves, training is discontinued. The parameter initializations for Step 1 are:

- Number of middle nodes: $H = 8$.
- Significance level for statistical testing of reduced structure models: $\alpha = .05$.
- Number of neural network runs: $I = 10$.

- Number of network features: $M = 4$.
- Number of training set exemplars: $P_{tr} = 181$.
- Number of training-test set exemplars: $P_{ts} = 100$.
- Total number of exemplars: $P = 281$.
- Number of network weight parameters: $s = 49$.

Ten architecture selection experiments are performed with different initial random number seeds. Backpropagation follows a unique gradient descent path for every run. The path is unique for three reasons: (1) the random partitioning of the training and test sets, (2) the random initialization of the weight vector, and (3) the random presentation of the training vectors. Therefore, the result of each experiment will be different. The results are summarized for the ten experiments in Table 23.

Table 23. Summary of Ten API Problem Architecture Selection Experiments

| Experiment Number | Final Number Middle Nodes | minimum $SSE$ | Corresponding % Classification Error |
|---|---|---|---|
| 1 | 8 | 4.398 | 1.779 % |
| 2 | 8 | 5.630 | 1.779 % |
| 3 | 7 | 6.373 | 2.850 % |
| 4 | 7 | 6.245 | 2.847 % |
| 5 | 7 | 6.445 | 2.135 % |
| 6 | 7 | 5.018 | 1.423 % |
| 7 | 6 | 4.656 | 2.491 % |
| 8 | 6 | 4.365 | 2.491 % |
| 9 | 6 | 5.003 | 2.847 % |
| 10 | 6 | 4.758 | 1.779 % |

These results demonstrate that backpropagation converges to a variety of local minimums. Each $SSE$ represents the smallest of the local minimums from 10 different runs. The 'local minimum' phenomena means the algorithm will not always converge to the most parsimonious architecture. However, these results seem to indicate the algorithm is conservative. Therefore, if the initial number of middle nodes is sufficient, the final number of middle nodes will also be sufficient, whether or not a reduction has taken place.

All of the selected architectures, whether or not they are reduced, are associated with *good* $SSE$'s and classification errors. On average, there is not a great difference between the minimum $SSE$'s or classification error rates of the various architectures as reported in Table 24. For the API problem, eight middle nodes are sufficient [6], and corresponds to the results of experiments one and two. In eight out of ten experiments, the selected architecture contains fewer middle nodes. To summarize, the algorithm selected a sufficient initial architecture for each experiment, and the sele:ted architecture was more *parsimonious* 80% of the time.

Table 24. API Problem: Average Performance of Selected Architectures

| Final Number Middle Nodes | Average Minimum $SSE$ | Corresponding % Classification Error |
|---|---|---|
| 8 | 5.014 | 1.779 % |
| 7 | 6.020 | 2.313 % |
| 6 | 4.696 | 2.402 % |

The tenth experiment which selected an architecture of six middle nodes is summarized in Table 25. Figure 17 shows the relationship between the number of model parameters and $SSE$ for each number of middle nodes tested. Similarly, Figure 18 shows the relationship between the number of model parameters and the total classification error for each number of middle nodes

Table 25. Summary of API Problem Architecture Selection (Experiment 10)

| Iteration | Number of Parameters | Number of Middle Nodes | Full $SSE$ | Reduced $SSE$ | Numerator degrees of freedom | Denominator degrees of freedom | Likelihood Ratio Statistic: $L$ | $F_{.05}$ Critical Value | Reject or Accept Reduced Model? |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 49/43 | 8/7 | 6.768 | 6.952 | 6 | 232 | 1.052 | 2.1 | Accept |
| 2 | 43/37 | 7/6 | 6.952 | 4.758 | 6 | 238 | -12.520 | 2.1 | Accept |
| 3 | 37/31 | 6/5 | 4.758 | 5.181 | 6 | 244 | 3.619 | 2.1 | Reject |

tested. The variation in local minimum for all 10 runs is displayed above the minimum $SSE$ run for each of the tested architectures. Although the selected architecture contained six middle nodes, the results for five middle nodes are shown for comparison. More detailed results of this experiment are shown in an audit log contained in Appendix B, Section B.1.

*5.3.5 Feature Selection Algorithm.* In this section, a sequential feature selection algorithm is proposed for examining candidate feature subsets for systematic elimination from a feature set. The algorithm can be implemented for feature sets which are larger than necessary which may contain irrelevant or redundant features. Alternatively, the algorithm can also be used to sequentially reduce a feature set which is larger than desired.

The algorithm uses the concepts of statistical model selection described in Section 2. Specifically, a backwards sequential selection procedure is used to systematically investigate reduced neural network models using likelihood ratio test statistics for ***statistically*** testing the reduced models. The feature selection algorithm is unique because initial architecture selection and architecture adjustment are embedded within the algorithm. Architecture adjustment allows the neural network to dynamically adapt its architecture as necessary throughout the feature selection process. Figure 19 depicts the neural network feature selection algorithm. A description of the algorithm follows.

Figure 17. API Problem Architecture Selection: *SSE* for 10 Runs (Experiment 10)

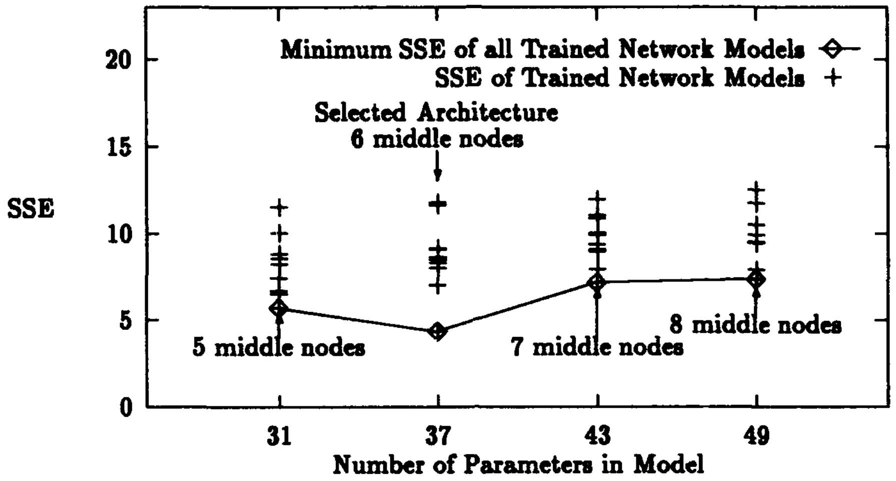


Figure 18. API Problem Architecture Selection: % Classification Error for 10 Runs (Experiment 10)

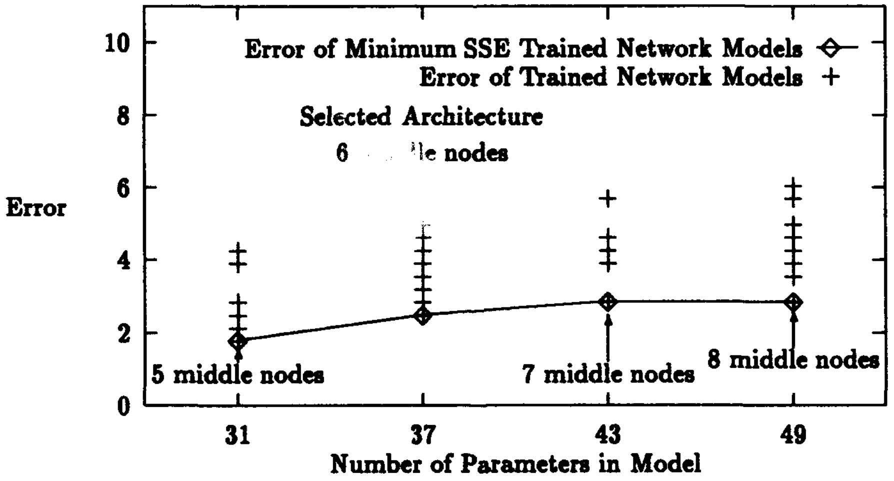
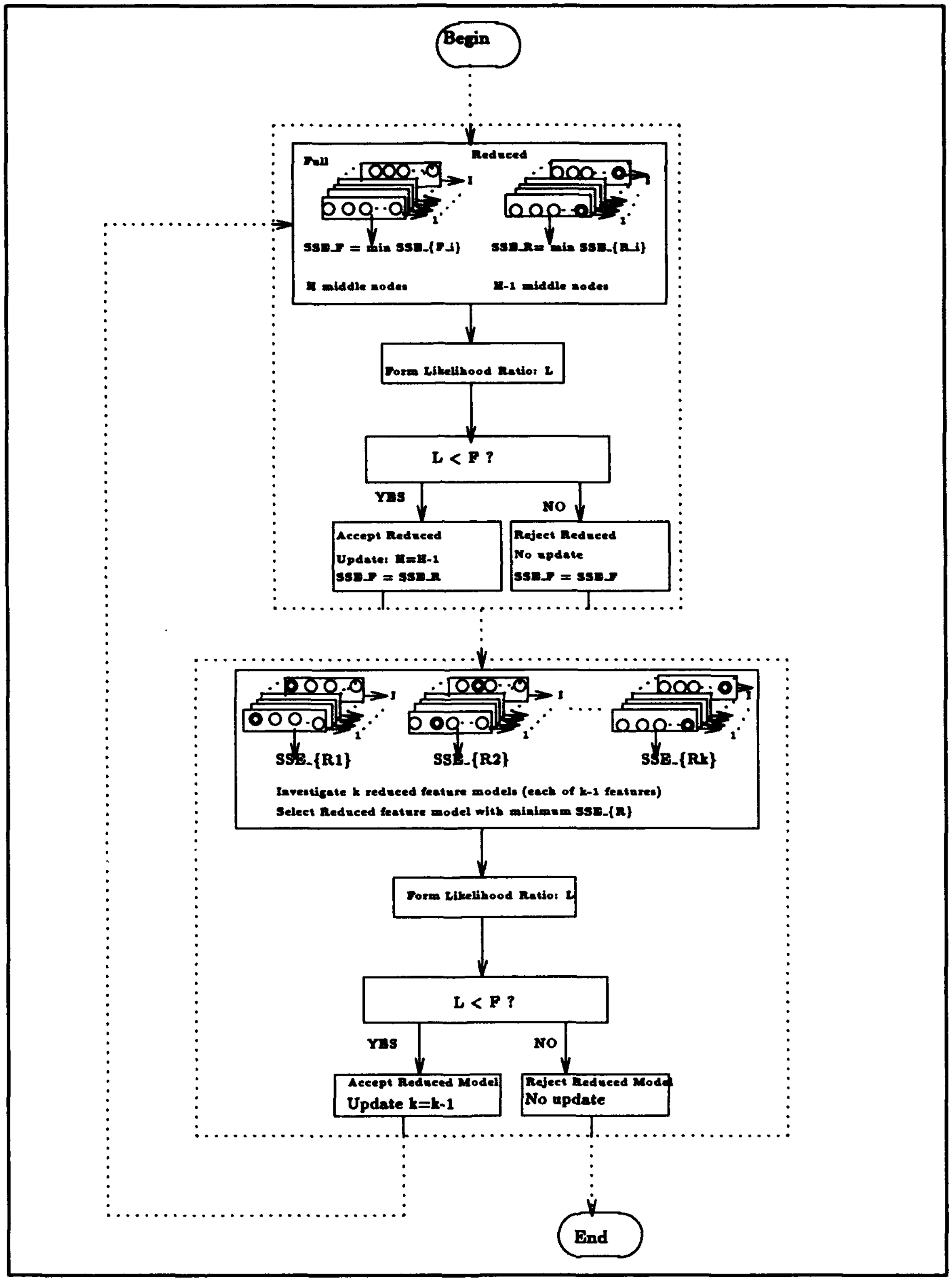


**Figure 19. Neural Network Feature Selection Algorithm**



## Neural Network Feature Selection Algorithm

1. **Initialize and define parameters.**

   - Number of features being evaluated: $M$
   - Current number of features being evaluated: $k$.

     Initialize $k = M$.
   - Current number of middle nodes: $H$.
   - Number of neural networks: $I$.
   - Number of training exemplars: $P_{tr}$.
   - Number of training-test exemplars: $P_{ts}$.
   - Total number of exemplars: $P = P_{tr} + P_{ts}$.
   - Number of neural network weight parameters: $s$.

     $s = (k+1)H + (H+1)$
   - Significance level for feature elimination: $\alpha_1$.
   - Significance level for middle node elimination: $\alpha_2$.
   - Set $f = 0$.

     If desired, $f$ can also be set equal to the final number of features to be selected.
   - Count=0.

2. Train $I$ realizations of the full neural network model with $k$ features and $H$ middle nodes.

   - Compute SSE using Equation 67.
   - Define $\text{SSE}_{F_i}$ = SSE for the $i$th neural network realization.
   - Define $\text{SSE}_F = \min\{\text{SSE}_{F_1}, \cdots, \text{SSE}_{F_I}\}$

3. Train $I$ realizations of the reduced neural network model with $k$ features and $H-1$ middle nodes.

    - Compute SSE using Equation 67.

    - Define $\text{SSE}_{R_i} = \text{SSE}$ for the $i$th neural network realization.

    - Define $\text{SSE}_R = \min\{\text{SSE}_{R_1}, \cdots, \text{SSE}_{R_I}\}$

4. Compute degrees of freedom for the full and reduced models.

    - $\text{df}_F = P - s$ (full model).

    - $\text{df}_R = P - [s - (k+2)]$ (reduced model).

5. Compute the Likelihood Ratio Test Statistic: $L$.

    - $L = \frac{[\text{SSE}_R - \text{SSE}_F]/(\text{df}_R - \text{df}_F)}{\text{SSE}_F/\text{df}_F}$.

6. Test the null hypothesis that the reduced model is equivalent to the full model using the likelihood ratio test statistic.

    - If $L \leq F_{\alpha_1, \text{df}_R - \text{df}_F, \text{df}_F}$, the reduced structure model can not be rejected.

        - Accept the reduced structure model (i.e. remove 1 middle node).

        - $H = H - 1$

        - $s = s - [k+1]$

        - $\text{SSE}_F = \text{SSE}_R$

        - $\text{df}_F = \text{df}_R$

        - If Count=1, then go to Step 2 and investigate a reduced architecture model before investigating feature reduction.

          If Count>1, then go to Step 7

    - If $L > F_{\alpha_1, \text{df}_R - \text{df}_F, \text{df}_F}$, then reject the reduced structure model.

  - $\mathrm{SSE}_F = \mathrm{SSE}_F$
  - $\mathrm{df}_F = \mathrm{df}_F$
  - Go to Step 7.

7. If $k = 1$ or $k \leq f$, then go to Step 13 since no further feature reduction is required.

8. Train $I$ realizations for each of the $k$ reduced feature candidate models. Each candidate model has $H$ middle nodes and a different feature set containing $k - 1$ of the $k$ remaining features which have not been previously eliminated.

   - Compute SSE using Equation 67.
   - Define $\mathrm{SSE}_{R_{ij}}$ = SSE for the $i$th neural network realization of the $j$th reduced feature model, where $j = 1, \cdots, k$.
   - Define $\mathrm{SSE}_R = \min\{\mathrm{SSE}_{R_{11}}, \cdots, \mathrm{SSE}_{R_{I1}}, \cdots, \mathrm{SSE}_{R_{1k}}, \cdots, \mathrm{SSE}_{R_{Ik}}\}$
   - Select as a candidate feature for elimination, the feature which when removed produces the model with the smallest $\mathrm{SSE}_R$.

9. Compute degrees of freedom for the reduced model.

   - $\mathrm{df}_R = P - [s - H]$ .

10. Compute the Likelihood Ratio Test Statistic: $L$.

    - $L = \frac{[\mathrm{SSE}_R - \mathrm{SSE}_F]/(\mathrm{df}_R - \mathrm{df}_F)}{\mathrm{SSE}_F/\mathrm{df}_F}$.

11. If $L < 0$ or $0 < f < k$ accept reduced feature model.

    - Eliminate the candidate feature.
    - $k = k - 1$
    - $s = s - H$

- Count=Count+1
- Go to Step 2.

12. Test the null hypothesis that the reduced model is equivalent to the full model using the likelihood ratio test statistic.

    - If $L \leq F_{\alpha_2, \mathrm{df}_R - \mathrm{df}_F, \mathrm{df}_F}$, the reduced model can not be rejected.
        - Accept the reduced feature model.
        - Eliminate the candidate feature.
        - $k = k - 1$
        - $s = s - H$
        - Count=Count+1
        - Go to Step 2.
    - If $L > F_{\alpha_2, \mathrm{df}_R - \mathrm{df}_F, \mathrm{df}_F}$ and $p = 0$, then reject the reduced feature model.
        - Go to Step 13.

13. Stop, the final neural network model has been determined with $H$ nodes and $k$ features.

In Step 1, parameters are initialized and defined. A reasonable number of neural networks is usually five or ten. When trying to find *good* or improved local minima, there is a point of diminishing returns for the computational cost of running additional networks.

In Steps 2 and 3, each training realization involves a different training and training-test set which is randomly selected from among the total pool of exemplars. Randomly selected training and training-test sets means that each neural network run is trained using a unique partitioning of the data.

To fairly compare results, the $SSE$ is computed in Steps 2 and 3 using all $P$ exemplars, so the full and reduced models are compared using the same data set. If there is sufficient data, a validation set $SSE$ can be used to provide the most unbiased comparison between models.

A finite number of full and reduced model networks are trained. From this, the networks corresponding to the best full and reduced model local minima are used to form the likelihood ratio in Step 5. In this scenario, where only a finite number of networks are trained, it is possible that the reduced model may produce a lower total $SSE$ than the full model. If the likelihood test statistic is negative in Step 6, the reduced model should be accepted, since the best reduced model from $I$ runs is better than the best full model from $I$ runs.

With any type of network redundancies (middle nodes or features), there are flats in the parameter space [80:106]. These flats can make convergence to a good local minimum more difficult; consequently, identification of an unnecessary or redundant feature may be more difficult. Therefore, initial architecture selection, as well as, architecture adjustment are incorporated into the algorithm.

Because the architecture selection is incorporated into the algorithm, middle node initialization is important (see Section 5.3.3). Architecture adjustment is investigated because, fewer input features may be associated with reduced network complexity. The algorithm attempts to maintain a minimal network structure at each step, thereby, allowing the focus to remain on the elimination of unnecessary features.

The feature selection algorithm is set up with two potential stopping criteria. The first stopping criteria is the most classical and is depicted in Figure 19. Here, the algorithm stops when the likelihood ratio test statistic $L$ is greater than the selected critical point of the $F$-distribution. If this stopping criteria is used, the parameter $f$ is set equal to zero.

The second stopping criteria involves stopping the algorithm after a predetermined number of features have been eliminated. In this case, the algorithm is implemented with the parameter

$f$ set equal to the final number of features to be selected. When this stopping criteria is used, the candidate feature for elimination is removed at each step as long as the current number of features is greater than the final number of feature desired, even when the likelihood ratio test statistic $L$ exceeds the critical point of the $F$-distribution. When the second stopping criteria is used, the following information should be reported at each step: SSE, $L$, P-values, classification error (for a classification problem), current features, and current number of middle nodes. With this information, a neural network practitioner can determine the most appropriate feature set by analyzing the tradeoffs between an accurate model and a parsimonious model.

It is ideal to do some screening and analysis prior to using the feature selection algorithm. Ideally, the *noise-like* features have been identified and removed using the feature screening techniques of Chapter IV.

*5.3.6 Application of the Feature Selection Algorithm.* In this section, the feature selection algorithm is applied to the FLIR problem introduced in Chapter III. For the FLIR results reported in Chapters III and IV, four middle nodes are used. Here, the FLIR problem is initialized with eight middle nodes to demonstrate the feature selection algorithm's effectiveness.

The results shown in Figure 10 of Chapter III, page 101 indicate that one to three features can be eliminated without a significant degradation in validation set results. Using the second stopping criteria, the feature selection algorithm is applied to select six good features with an appropriate network architecture. Five neural networks are used in this experiment. Each network is trained for a minimum of 500 epochs and improvement in the test set $SSE$ is monitored every 50 epochs to justify continuation of training. Once the test set $SSE$ no longer improves, training is discontinued. The parameter initializations for Step 1 are:

- Number of middle nodes: $H = 8$.
- Significance level for statistical testing of reduced structure models: $\alpha = .05$.

- Significance level for statistical testing of reduced feature models: $\alpha = .05$.
- Number of neural network runs: $I = 5$.
- Number of network features: $M = 9$.
- Number of training set exemplars: $P_{tr} = 350$.
- Number of training-test set exemplars: $P_{ts} = 200$.
- Total number of exemplars: $P = 550$.
- Number of network weight parameters: $s = 89$.
- Final number of features to be selected: $f = 6$.

The results from applying the feature selection algorithm to the FLIR problem are summarized in Table 26. For the reduced models accepted within each iteration of the algorithm, Figure 20 shows the relationship between the number of model parameters and minimum total $SSE$. Similarly, Figure 21 shows the relationship between the number of model parameters and the total classification error for each of the accepted models.

Both the first and second stopping rules are indicated in Figures 20 and 21. Using the first stopping rule, only ***statistically*** equivalent reduced models are selected. The indicated stopping point corresponds to a 43% reduction in parameters with no degradation in accuracy. The second stopping rule allows the practitioner the flexibility to trade off accuracy for a more parsimonious model. In this case, the indicated stopping point corresponds to a 63% reduction in parameters with only a small degradation in accuracy. Notice that using the second stopping rule uncovers three additional models for consideration. The best of these additional models occurs with four middle nodes and seven features. More detailed results of this experiment are shown in an audit log contained in Appendix B, Section B.2.

Since saliency metric results in Figure 10 indicate that one to three features can be eliminated, the six most salient features would be a logical alternative for selecting six good features. However,

Figure 20. FLIR Problem Feature Selection: Minimum $SSE$'s

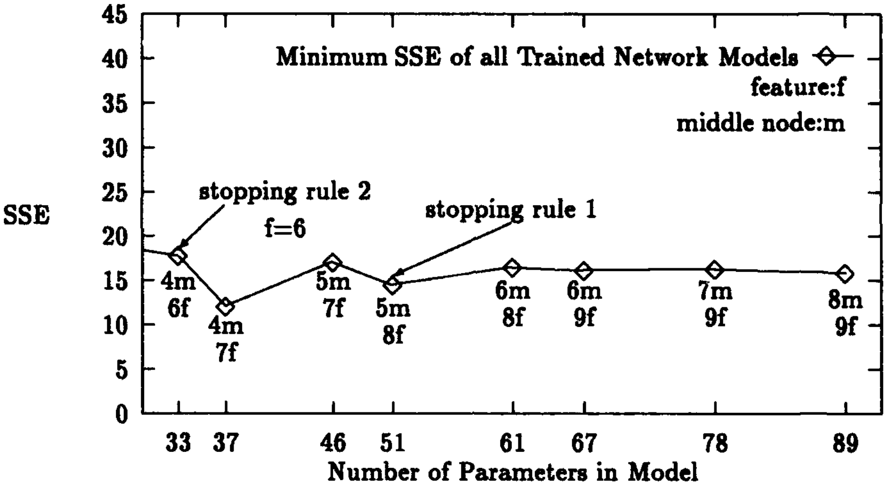


Figure 21. FLIR Problem Feature Selection Experiment: % Classification Error for Minimum $SSE$ Networks

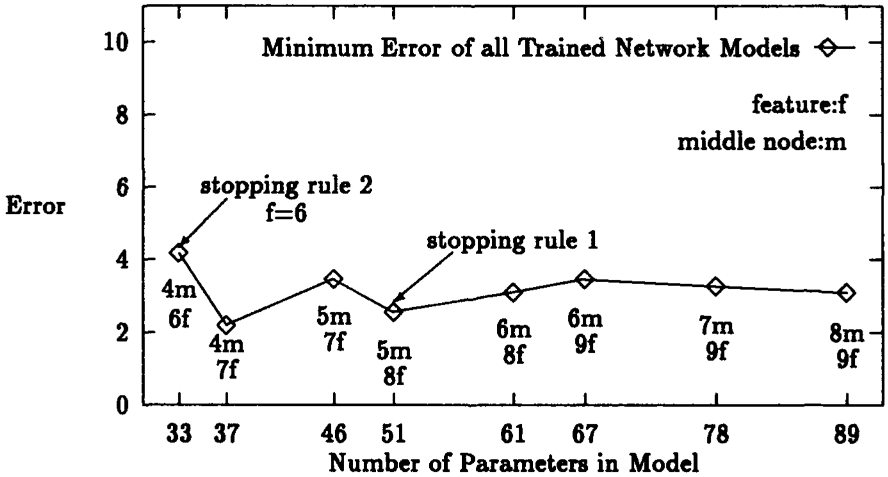


Table 26. Summary of a Feature Selection Experiment for the FLIR Problem

| Iteration | Number of Parameters | Number of Middle Nodes | Number of features | Full $SSE$ | Reduced $SSE$ | Numerator and Denominator degrees of freedom | Likelihood Ratio Statistic: $L$ | $F_{.05}$ Critical Value | Reject or Accept Reduced Model? |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 89/78 | 8/7 | 9 | 15.906 | 16.327 | 11 and 461 | 1.109 | 1.79 | Accept: $L < F_\alpha$ |
| 1 | 78/67 | 7/6 | 9 | 16.327 | 16.139 | 11 and 472 | -0.494 | 1.79 | Accept: $L < F_\alpha$ |
| 1 | 67/56 | 6/5 | 9 | 16.139 | 19.724 | 11 and 483 | 9.754 | 1.79 | Reject: $L > F_\alpha$ |
| 1 | 67/61 | 6 | 9/8 | 16.139 | 16.554 | 6 and 483 | 2.067 | 2.10 | Accept: $L < F_\alpha$ |
| 2 | 61/51 | 6/5 | 8 | 16.554 | 14.553 | 10 and 489 | -5.911 | 1.83 | Accept: $L < F_\alpha$ |
| 2 | 51/46 | 5 | 8/7 | 14.553 | 17.071 | 5 and 499 | 17.27 | 2.21 | Accept: $(f < 8)$ |
| 3 | 46/57 | 5/4 | 7 | 17.071 | 12.099 | 9 and 504 | -16.310 | 1.88 | Accept: $L < F_\alpha$ |
| 3 | 57/53 | 4 | 7/6 | 12.099 | 17.845 | 4 and 513 | 60.91 | 2.60 | Accept: $(f < 7)$ |
| 4 | 53/45 | 4/3 | 6 | 17.845 | 19.495 | 8 and 517 | 5.974 | 1.94 | Reject: $L > F_\alpha$ |

eliminating the three least salient features does not take into account that a seemingly unimportant feature may be interacting in an important way with the remaining features. Whereas, the sequential feature selection algorithm described in this section partially takes the feature interactions into account when selecting a feature subset.

Indeed, the feature selection algorithm does not select the three least important features for elimination. Out of the nine FLIR features, the 3rd, 6th and 8th ranked salient features using $\hat{\Lambda}_i^{\text{data}}$ in Table 12 on page 98 are eliminated using the feature selection procedure. Figures 22 and 23 can be studied to compare the error rates over 800 epochs of training for the 'selected feature subset' versus the 'salient feature subset.' The 'salient feature subset' contains the six most salient features using the saliency metric $\hat{\Lambda}_i^{\text{data}}$.

In Figures 22 and 23, the minimum $SSE$ and the corresponding percent classification error from 10 runs are plotted for the $P$ vectors used in the feature selection experiment. The results

Figure 22. FLIR Problem Feature Subsets: Minimum *SSE*

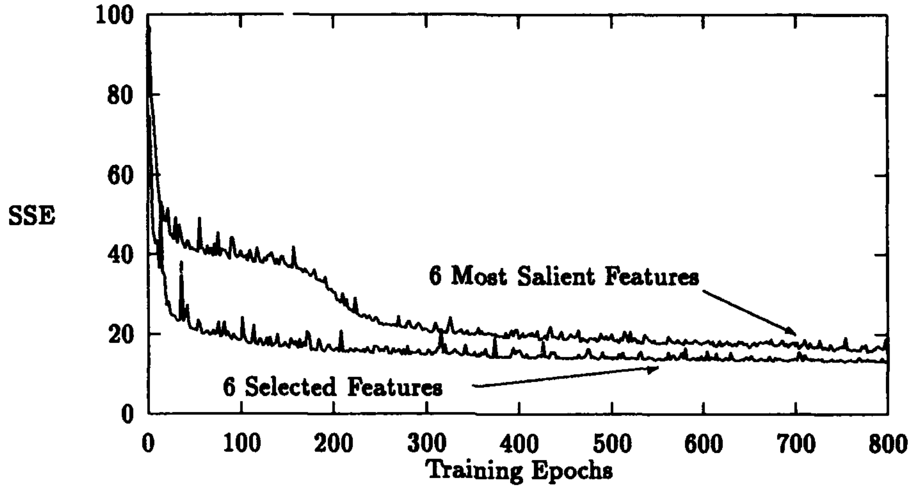
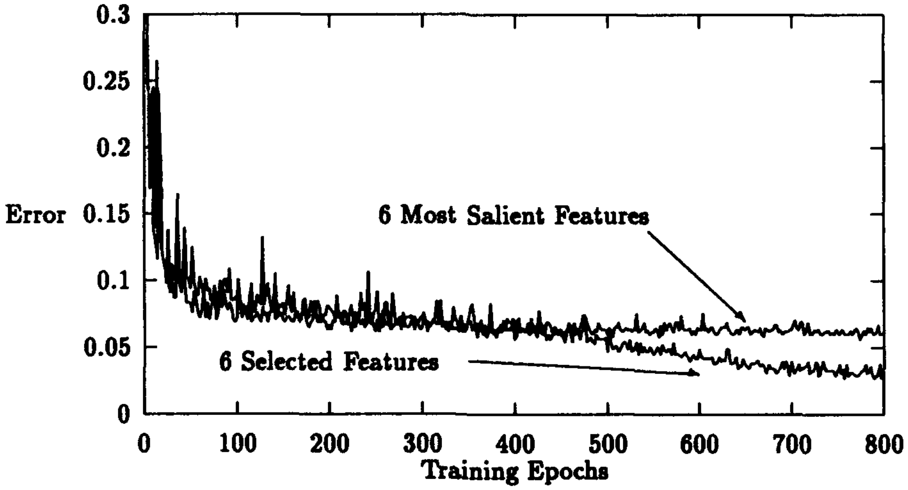


Figure 23. FLIR Problem Feature Subsets: % Classification Error for Minimum *SSE* Network



shown for the minimum $SSE$ network tell the same story as the average results over the 10 runs. The average $SSE$ for the selected features versus the most salient features was 24.58 and 30.36, respectively at 800 epochs. Similarly, the average percent classification error for the selected features versus the most salient features was 5.23 and 6.96, respectively at 800 epochs. The average percent classification error is significantly better (at the 0.05 statistical significance level) for the subset of selected features. This illustrates a quantifiable advantage to using the selection procedure on the FLIR problem.

### *5.4 Summary*

Since a neural network can be viewed as a nonlinear regression model, nonlinear regression model selection concepts are applicable for neural networks. This chapter reviews nonlinear regression model selection and the practical considerations encountered in a neural network model selection scenario. Nonlinear regression model selection is the basis for proposing architecture and feature selection algorithms for neural networks. Both algorithms use the likelihood ratio test statistic (shown in Equation 70 on page 143) within a backwards sequential selection procedure.

The first algorithm is an architecture selection algorithm for determining a good number of middle nodes. The algorithm is used to automate what is often a process of trial and error experimentation for many practitioners. Application results from ten initial architecture selection experiments using the API problem data are presented. A reduced architecture was determined for this problem for eight of the ten experiments.

The second algorithm is a feature selection algorithm for statistically investigating reduced feature subsets. Embedded in the algorithm is initial architecture selection and architecture adjustment. The network architecture is dynamically adapted as necessary throughout the feature selection process. Application results from a feature selection experiment on the FLIR problem demonstrate that 43% and a 63% reduction of parameters for using two different stopping rules.

The 43% parameter reduction produced approximately the same prediction accuracy. This reduction corresponded to accepting only *statistically* equivalent reduced models using the likelihood ratio test statistic. The 63% parameter reduction produced a slightly degraded prediction accuracy, and it corresponded to stopping the algorithm after three features had been eliminated, whether or not the reduced model was *statistically* equivalent. When the six most salient features are compared to the 'selected feature subset,' the prediction accuracy from the 'salient feature subset' is significantly lower than the 'selected feature subset' at the .05 statistical significance level. This result is possible because the feature selection algorithm partly takes the correlation structure of the features into account.

In the next chapter, a comprehensive neural network selection methodology is developed for identifying both a good feature set and an appropriate neural network architecture for a specific situation. The methodology combines both the statistical screening procedure from Chapter IV and the statistical architecture and feature selection algorithms into a comprehensive statistically-based approach for neural network model selection.

# *VI. A Comprehensive Neural Network Selection Methodology*

## *6.1 Introduction*

In Chapter IV, a statistical screening procedure is developed for the identification of *noise-like* irrelevant features using the saliency metrics developed in Chapter III. In Chapter V, neural network selection algorithms for architecture selection and feature selection are proposed. In this chapter, a comprehensive neural network selection methodology is presented which logically integrates the screening and selection algorithms into an overall statistically-based approach for neural network model selection. As in the previous chapters, application of the proposed methodology is investigated using a single hidden layer architecture with sigmoidal activation function on the middle and output layers. However, this methodology is general and is, therefore, appropriate for any feedforward neural network architecture using a variety of activation functions. Changes in architecture or activation functions would merely change the definitions of the neural network error function which is minimized. This in-turn changes the definitions of the saliency function described in Chapter III.

## *6.2 Comprehensive Neural Network Selection Methodology*

*6.2.1 Introduction.* The methodology proposed in this section is the culmination of this dissertation effort. The proposed methodology represents the logical combination of research results from Chapters III, IV, and V into an overall procedure for dynamically identifying both a good feature set and an appropriate neural network structure for a specific situation.

*6.2.2 Comprehensive Selection Methodology.* The comprehensive selection methodology depicted in Figure 24 is composed of three separate modules within the algorithm: (1) initial architecture selection, (2) saliency screening, and (3) feature selection.

The initial architecture module is described in Section 3 of Chapter V. The saliency screening module is described in Section 2 of Chapter IV. The feature selection module is described in Section 3 of Chapter V. A description of the comprehensive neural network selection methodology is depicted in Figure 24 follows.

### Comprehensive Neural Network Selection Methodology

1. Initialize and define parameters for:

   - Initial architecture selection algorithm.
   - Saliency screening procedure.
   - Feature selection algorithm.

2. Perform initial architecture selection algorithm.

3. Perform saliency screening and remove any *noise-like* features which are identified.

4. Perform feature selection algorithm.

   When no additional features can be removed based on the stopping rules, continue to Step 5.

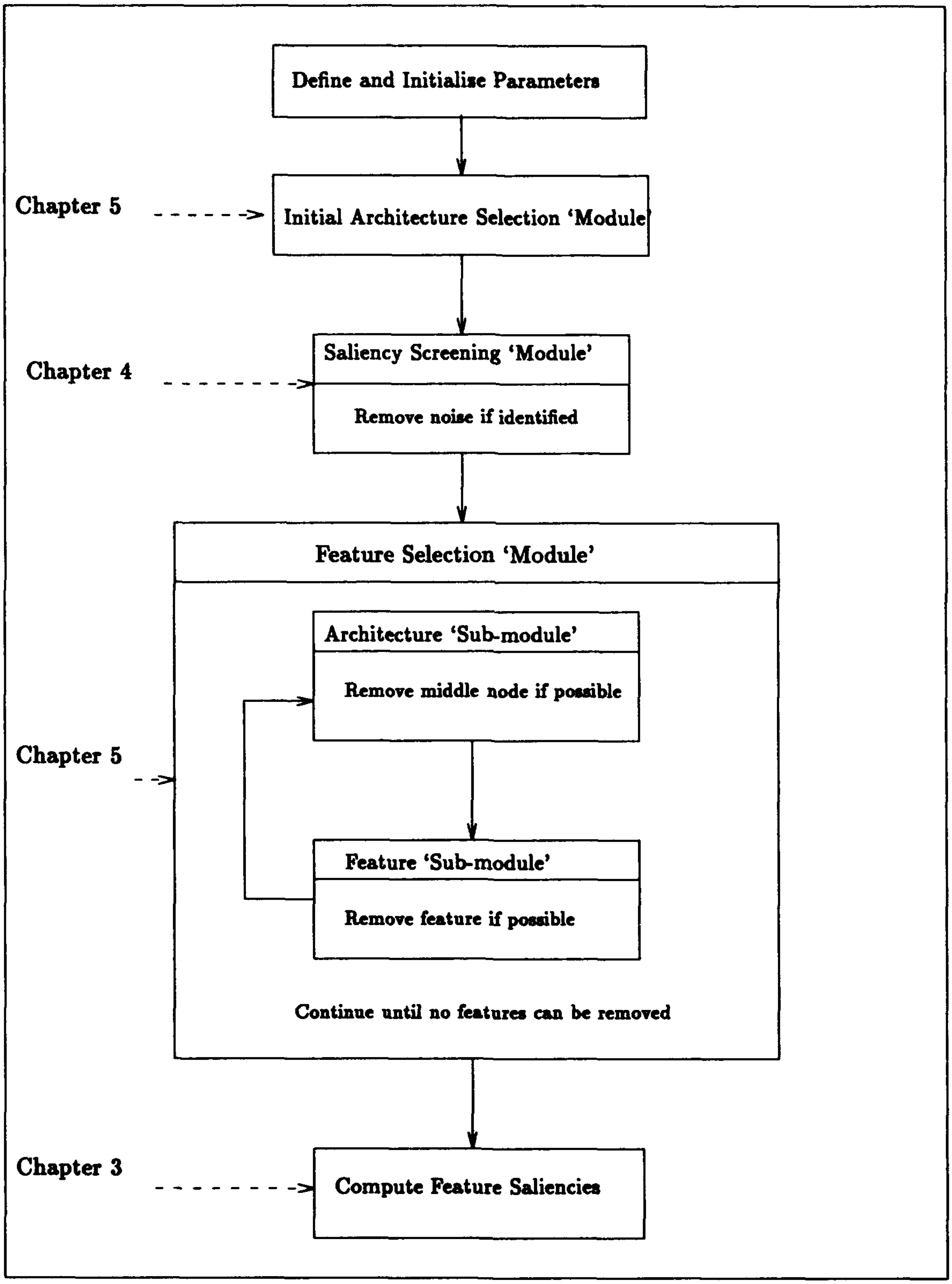
5. Stop, a final neural network has been selected for the problem at hand.

   Compute the associated feature saliencies.

The comprehensive selection methodology calls for finding an appropriate initial neural network architecture, and then screening and eliminating any *noise-like* features. Next, the methodology calls for iteratively investigating the elimination of unnecessary or redundant middle nodes and features. When no further features can be removed, the saliencies are calculated for the remaining features using the final network architecture.

With any type of network redundancies (middle nodes or features), there are flats in the parameter space [80:106]. These flats can make convergence to a good local minimum more difficult; consequently, identification of an unnecessary or redundant feature may be more difficult.

Figure 24. Comprehensive Neural Network Selection Methodology


Define and Initialise Parameters
Chapter 5
Initial Architecture Selection 'Module'
Chapter 4
Saliency Screening 'Module'
Remove noise if identified
Feature Selection 'Module'
Architecture 'Sub-module'
Remove middle node if possible
Chapter 5
Feature 'Sub-module'
Remove feature if possible
Continue until no features can be removed
Chapter 3
Compute Feature Saliencies

Therefore, as unnecessary features are removed, the network structure is dynamically reduced as necessary. The algorithm attempts to maintain a minimal network structure at each step, thereby, allowing the algorithm to focus on the elimination of unnecessary features.

*6.2.3 Scope of Application* Two issues are discussed in this section. First the scope of the algorithm's application is discussed. Second, the prospects for parallel processing are discussed.

Due to the multi-module/multi-iteration characteristics of the proposed network selection algorithm, it is most efficient and prudent for problems where the total number of candidate features is a manageable number of at most 20 or 30 features. This means the practitioner may need to apply screening, intuition, or feature rotations to reach a reasonable number of features before using the selection methodology.

The multi-iteration characteristic of the proposed network selection algorithm, makes the algorithm amenable to parallel computation. For the architecture and feature modules, multiple iterations are used to ensure that a good local minimum is found. All of these iterations could be parsed out and trained on separate networks; subsequently, the results from the trained networks could be centrally analyzed. In the saliency screening module, multiple iterations are used so saliency statistics can be collected for statistical hypothesis testing. Similar to the architecture and feature modules, the iterations could be parsed out and trained on separate networks, and the statistics could be centrally collected and analyzed.

*6.2.4 Application of the Comprehensive Selection Algorithm.* To test the comprehensive selection algorithm, the XOR problem was modified to include the following features: $x$, $y$, $n_1$, $n_2$, and $m_1$. The true features shown in Figure 7, page 90 are $x$ and $y$; $n_1$ and $n_2$ are augmented noise features; and $m_1$ is an augmented feature which is highly correlated with $x$. The feature $m_1$ is defined

$$m_1 = x + 0.01 \text{ UNF}(0,1)$$

The modified $XOR$ problem was designed to demonstrate that the saliency screening could be used to identify and eliminate one or both of the noise features, and that the feature module could be used to identify and eliminate either $x$ or $m_1$, since the presence of both features constitutes redundant feature information.

The selection experiment is performed with five network runs for the architecture and feature modules, and ten network runs for the saliency modules. In the feature selection module, the first stopping criteria is used. This means the algorithm stops when no additional features can be removed using the likelihood ratio test statistics. The networks are trained for a minimum of 500 epochs and improvement in the test set $SSE$ is monitored every 50 epochs to justify continuation of training. Once the test set $SSE$ no longer improves, training is discontinued. The parameter initializations for Step 1 are:

- Number of middle nodes: $H = 6$.
- Significance level for statistical testing of reduced structure models: $\alpha_1 = .05$.
- Significance level for statistical testing of reduced feature models: $\alpha_2 = .05$.
- Individual significance level for saliency screening for *noise-like* features: $\alpha_3 = .01$ (family significance level of 0.05)
- Number of architecture and feature module neural network runs: 5.
- Number of saliency module neural network runs: 10.
- Number of network features: $M = 5$.
- Number of training set exemplars: $P_{tr} = 300$.
- Number of training-test set exemplars: $P_{ts} = 200$.
- Total number of exemplars: $P = 500$.
- Number of network weight parameters: $s = 43$.

Table 27. Selection Experiment with the Modified XOR Problem

| Module | Number of Parameters | Number of Middle Nodes | Number of features | Full $SSE$ | Reduced $SSE$ | Numerator and Denominator degrees of freedom | Likelihood Ratio Statistic: $L$ | $F_{.05}$ Critical Value | Reject or Accept Reduced Model? |
|---|---|---|---|---|---|---|---|---|---|
| Init Arch | 43/36 | 6/5 | 5 | 7.912 | 8.091 | 7 and 457 | 1.475 | 2.01 | Accept: $L < F_\alpha$ |
| Init Arch | 36/29 | 5/4 | 5 | 8.091 | 8.338 | 7 and 464 | 2.026 | 2.01 | Reject: $L > F_\alpha$ |
| Sal Screen | 36/26 | 5 | 3 | n/a | n/a | n/a | n/a | n/a | Accept |
| Feature | 26/21 | 5/4 | 3 | 7.643 | 7.385 | 5 and 464 | -3.126 | 2.21 | Accept: $L < F_\alpha$ |
| Feature | 21/16 | 4/3 | 3 | 7.385 | 16.222 | 5 and 469 | 112.232 | 2.21 | Reject: $L > F_\alpha$ |
| Feature | 21/17 | 4 | 3/2 | 7.385 | 6.496 | 4 and 469 | -14.122 | 2.37 | Accept: $L < F_\alpha$ |
| Feature | 17/13 | 4/3 | 2 | 6.496 | 16.428 | 4 and 473 | 180.808 | 2.37 | Reject: $L > F_\alpha$ |
| Feature | 17/13 | 4 | 2/1 | 6.496 | 124.431 | 4 and 473 | 2146.919 | 2.37 | Reject: $L > F_\alpha$<br>Stop |

Table 27 summarizes the experiment. The saliency screening procedure is separately summarized in Table 28, since the format of Table 27 is not adequate. The initial architecture selection module reduces the network architecture from six middle nodes to five middle nodes. The saliency screening module identifies both $n_1$ and $n_2$ as *noise-like*, which reduces the network model to five middle nodes and three features. In the feature module, the architecture is reduced to four middle nodes, and the reduced feature model with $y$ and $m_1$ is identified as equivalent to the full feature model with $x$, $y$, and $m_1$. In summary, the comprehensive feature selection procedure is used to sequentially reduce the neural network model from 43 parameters to 17 parameters using likelihood ratio test statistics to sequentially identify reduced models which are approximately *statistically* equivalent. This represents a 60% reduction in weight parameters, with no degradation in prediction or classification accuracy.

More detailed results of this experiment are shown in an audit log contained in Appendix B, Section B.3.

Table 28. Saliency Screening Results on XOR Problem for 10 Runs

| Feature $i$ | $\bar{\Lambda}_i^{\text{data}}$ Ranking | $\bar{\Lambda}_i^{\text{data}}$ | $\sigma_{\bar{\Lambda}_i^{\text{data}}}$ | $\bar{D}_i$ | $S_{\bar{D}_i}$ | $t^* = \frac{\bar{D}_i}{S_{\bar{D}_i}}$ | $\nu$ | Reject $H_0$ |
|---|---|---|---|---|---|---|---|---|
| $x$ | 2 | 0.558 | 0.703E-01 | 0.455 | 0.746E-01 | 6.09 | 9 | yes |
| $y$ | 1 | 1.036 | 0.856E-01 | 0.933 | 0.910E-01 | 10.3 | 9 | yes |
| $n_1$ | 4 | 0.107 | 0.317E-01 | 0.348E-02 | 0.371E-01 | 0.939E-01 | 9 | no |
| $n_2$ | 6 | 0.073 | 0.229E-01 | -0.301E-01 | 0.223E-01 | -1.35 | 9 | no |
| $m_1$ | 3 | 0.532 | 0.443E-01 | 0.429 | 0.301E-01 | 14.2 | 9 | yes |
| $x_n$ | 5 | 0.103 | 0.365E-01 | N/A | N/A | N/A | N/A | N/A |

Features ranked from best to worst

Individual Significance Level $\frac{\alpha}{M} = .01$

Bonferroni Critical Value $B = 2.821$

### *6.3 Summary.*

The comprehensive selection methodology proposed in this Chapter represents the logical integration of the statistical screening and selection procedures of Chapters IV and V, along with the saliency metrics of Chapter III. The methodology is designed for identifying both a good feature set and a potentially reduced neural network architecture for a specific situation. The scope of application and the potential for parallel implementation are discussed. An application experiment on a modified XOR problem demonstrate the utility of the methodology for identifying and eliminating both *noise-like* and redundant features, while adjusting the neural network architecture as necessary. In Chapter VII, the dissertation research is summarized and future research recommendations are made.

# *VII. Summary and Recommendations*

## *7.1 Introduction*

Feature and model selection for feedforward neural networks are advanced in this research. Feature selection involves determining a good feature subset from a set of candidate features, and the process of feature selection is characterized by three components in this research: (1) a metric or criterion function for evaluating and ranking the features, (2) a procedure for identifying irrelevant and redundant features, and (3) a search methodology for examining candidate feature subsets. Model selection involves determining an appropriate architecture (number of middle nodes) for the neural network. In this chapter, contributions advancing the process of neural network feature and model selection are summarized and recommendations for future research are made.

## *7.2 Summary*

This research begins with a review of feature selection techniques developed for regression, discriminant analysis, and neural networks. Because neural network applications include problems which have often been solved using classical regression and discriminant analysis techniques, non-neural feature selection techniques are reviewed for their potential use in a neural network context. What follows is a summary of the research advances and contributions made in the areas of feature saliency, identification of noisy features, and neural network model selection.

*7.2.1 Feature Saliency Metrics.* Feature saliency metrics are used to measure the relative importance of a feature with respect to a trained neural network. This research consolidates the set of available neural network feature saliency metrics by developing a catalogue of feature saliency metric definitions and interrelationships.

In this research, a framework is developed and used for analyzing a variety of derivative-based metrics. Several of the derivative-based saliency metrics are evaluated for their sensitivities to sampling, training, and redundant middle nodes. The metrics do not appear to be particularly sensitive to sampling; however, they are sensitive to redundant middle nodes and the amount of training. It is most important to eliminate redundant middle nodes, since the metrics are most sensitive to training effects in the presence of redundant middle nodes. Increased training may cause the weights associated with the redundant nodes to grow disproportionately. This can contaminate saliency results, since the weights from irrelevant features can get large.

A theoretical relationship is shown between derivative-based and weight-based saliency. In summary, the derivative-based feature saliency metrics are bounded above by a constant linear combination of the feature weights. At one middle node, the 'saliency function' ratios produced with the weight-based metrics and derivative-based metrics are equal, to within roundoff error. When additional middle nodes are used, empirical results indicate that the relative saliency of important to unimportant features is smaller with weight-based saliency than it is for derivative-based saliency. For problems with redundant middle nodes, this is partly due to the growth of the irrelevant weights associated with redundant middle nodes.

Contributions are also made in the area of Bayesian-based feature saliency metrics. First, a succinct and exact relationship is demonstrated between a previously suggested Bayesian-based metric and derivative-based saliency. Then a new Bayesian-based saliency metric using the partial derivative of classifier error is introduced. The computation of this metric requires only a subset of the terms associated with the previously suggested Bayesian-based metric. Finally, the relationship between the new Bayesian-based saliency and derivative-based saliency is derived, and an upper bound for the Bayesian-based saliency is defined. For a two class problem, the new metric produces results exactly equivalent to derivative-based saliency.

The catalogue of metrics are evaluated for a 'real world' problem. On this problem, saliency rankings, 'saliency function' ratios, and factor analysis are used to empirically evaluate similarities and differences between the saliency metrics. One similarity is that, despite differences, all of the metrics consistently ranked a set of 'nonessential' features last. Since the metrics perform differently, recommendations are made for selecting a feature saliency metric.

For discriminant analysis problems using networks with more than one middle node, the new saliency metric $\Gamma_i$ in Equation 53 on page 86 is preferable to $\hat{\Lambda}_i^{\text{data}}$ in Equation 31 on page 68 for two reasons. First, it is intuitively appealing, since it represents a saliency metric which is related to the average classifier $P_e$. Second, it is more succinctly defined using only a subset of the terms required for computation of $\hat{\Lambda}_i^{\text{data}}$.

For function approximation or discriminant analysis problems using networks with more than one middle node, the proposed saliency metric $\hat{\Lambda}_i^{\text{data}}$ in Equation 31 on page 68 should be preferred over $\hat{\Lambda}_i$. The saliency metric $\hat{\Lambda}_i^{\text{data}}$ provides good feature rankings, is more succinctly defined than $\hat{\Lambda}_i$, and is based on information known about the data from feature space.

For any classification or function approximation problem using a network with only one middle node, the weight-based metrics $\Upsilon_i^r$ (see Equation 32 on page 78) are best. In this case, the relative saliencies produced using weight-based saliency will be identical to $\Gamma_i$ or $\hat{\Lambda}_i^{\text{data}}$. For networks using more than one middle node, weight-based saliency can still be used for a cursory analysis of a feature's relative importance. However, the empirical results suggest that the relative importance of one feature to another is degraded when additional middle nodes are used.

*7.2.2 Identification of Noisy Features.* A saliency screening procedure for identifying noisy features is developed based on statistically comparing the mean saliency of candidate features to the mean saliency of a noisy feature. This research extends the work of Belue and Bauer to jointly screen an entire feature set for irrelevant features using a paired-$t$ hypothesis test. Irrelevant features are successfully identified over a series of test problems using the proposed saliency screening procedure. The procedure is robust for identifying irrelevant features, even in the presence of input redundancies.

The saliency screening technique is compared to the irrelevant input hypothesis test proposed by White [79]. The same test problems are used to evaluate the weight screening procedure. On any single run, the weight screening results were not reliable. When average test statistics from several runs are used, the weight screening procedure does provide comparable results for two of the three problems.

*7.2.3 Neural Network Model Selection.* Two novel neural network selection algorithms are developed by posing the neural network model as a nonlinear regression statistical model. Both algorithms are developed using the likelihood ratio test statistic (shown in Equation 70 on page 143) within a backwards sequential selection procedure. The first algorithm is an architecture selection algorithm which automates the process of determining an appropriate number of middle nodes. The second is a feature selection algorithm for statistically investigating reduced feature subsets. The feature selection algorithm is unique because architecture reduction is investigated as features are removed. The feature selection algorithm indirectly takes the correlation structure of the features into account. For this reason, a better reduced feature set may be identified using the feature selection algorithm versus using a subset of the most salient features. Application results demonstrate how these algorithms can be used to search for a more parsimonious neural network model with equivalent prediction accuracy.

A comprehensive neural network selection methodology is developed for identifying both a good feature set and an appropriate neural network structure for a specific situation. It encompasses a combination of statistical screening and statistical architecture and feature selection. Application results demonstrate how the comprehensive methodology can be used for identifying and eliminating both *noise-like* and redundant features, as well as reducing the number of middle nodes in the neural network architecture.

## *7.3 Recommendations*

There are related research topics which could not be adequately handled within this research effort. Four of the research topics could be pursued with worthwhile benefits.

The first research topic is an extension of the model selection procedures in Chapters V and VI. These single output procedures need to be extended for a multi-output neural network. A covariance adjusted sum of squared-errors must be computed for the multivariate response likelihood ratio test statistic. This involves separately estimating the covariance matrix of the neural network responses.

The second research topic is the development of an automatic stopping point selection for neural network training runs. The procedures proposed in Chapters IV, V, and VI require multiple runs of trained neural networks. Normally, trained networks correspond to a minimum training-test set error. However, a trained network is difficult to automatically identify within a multiple run procedure for several reasons. First, a network may take fewer or greater epochs, from run to run, depending on the gradient descent path taken with the backpropagation algorithm. Second, there is a great deal of variability in a network's error during training. Third, there may be several epochs where the test set error does not improve prior to eventual convergence. Fourth, neural networks may converge to a variety of local minima, to saddle points, or even diverge, making it unrealistic to expect all networks to reach a predetermined target value.

The model selection algorithms described in Chapter V, depend on the network's $SSE$ which can be biased low or high depending on where the stopping point is selected. In this research, prior knowledge was used to select a minimum number of training epochs, and then the network was trained as long as periodic monitoring indicated a reduction in the test set error. This procedure should be improved. A relatively simple yet accurate method for automatic stopping point selection is needed for practical implementation within the multiple runs procedures proposed in this dissertation.

The third research topic is the development of a diagnostic tool for identifying neural network redundancies. Throughout this research, feature space and middle node redundancies were an issue. There is a great need for a statistical or non-statistical technique for reliably diagnosing the existence of these redundancies in neural networks. The specification robust irrelevant input hypothesis can only be reliably used in the absence of these redundancies [80]. If these redundancies are difficult to diagnose and remedy a priori, then procedures are needed which are robust to the possible presence of these redundancies.

The fourth research topic is the development of residual analysis techniques for a neural network practitioner. There is a dearth of information available on residual analysis of neural network models. Residual analysis is well documented in a regression setting for examining the model aptness, but has not been documented in a neural network setting. Although neural network residuals are not generally normally distributed, residual analysis techniques should be developed which would be of practical benefit to a practitioner.

# Appendix A. *Derivation and Details of Two Saliency Metrics*

## *A.1 Second Order Feature Evaluation Metric*

Le Cun and others introduced a technique called Optimal Brain Damage for reducing the size of a neural network by selectively deleting weights or units from a neural network based on their saliencies [39]. The technique can be applied to feature evaluation when all the weight parameters from a feature are evaluated together.

Le Cun and others approximate the error of a neural network using a Taylor's series expansion, where neural network error, $\mathcal{E}_o$, refers to a function of the squared error [39]. A perturbation of the weight parameters connected to feature $i$ will change the neural network error by

$$d\mathcal{E}_o = \sum_{j=1}^{H} g_j dw^1_{ij} + \frac{1}{2}\sum_{j=1}^{H} h_{jj}(dw^1_{ij})^2 + \frac{1}{2}\sum_{j=1}^{H}\sum_{k\neq j}^{J} h_{jk} dw^1_{ij} dw^1_{ik} + O(\|dw^1_{ij}\|^3), \qquad (74)$$

where

$$g_j = \frac{\partial\mathcal{E}_o}{\partial w^1_{ij}} \quad \text{and} \quad h_{jk} = \frac{\partial^2\mathcal{E}_o}{\partial w^1_{ij}\partial w^1_{ik}}$$

This saliency metric measures the impact to $\mathcal{E}_o$ when deleting the feature to middle node weights for each feature. When a non-salient feature's associated weights are deleted, the change in $\mathcal{E}_o$ is small. When a highly salient feature's associated weights are deleted, the change in $\mathcal{E}_o$ is large. In order to make it computationally practical to evaluate the Taylor series expression for the change in error, Le Cun and others make three simplifying assumptions [39]:

1. They assume they are at a local minimum of the error which makes the first order terms equal to zero.

2. They make a diagonalizing assumption to eliminate cross terms.

3. They make a quadratic approximation assumption which implies that 3rd order and higher order terms are negligible.

What remains is the second order terms, which should be positive at a local minimum. This means any change in error due to perturbation of a parameter will be an increase in error. The simplifying assumptions reduce Equation 74 to

$$d\mathcal{E}_o = \frac{1}{2}\sum_{j=1}^{H} h_{jj}(d\,w_{ij}^1)^2, \tag{75}$$

The neural network expression for $\mathcal{E}_o$ is

$$\mathcal{E}_o = \frac{1}{2}\sum_{k=1}^{K}[d_k - z_k]^2$$

To show the detailed neural network notation for Equation 75, the diagonal of the Hessian $h_{jj}$ must be derived in detail:

$$\begin{aligned}
h_{jj} &= \frac{\partial^2 \mathcal{E}_o}{(\partial w_{ij}^1)^2} \\
&= \frac{\partial}{\partial w_{ij}^1}\left(\frac{\partial \mathcal{E}_o}{\partial w_{ij}^1}\right) \\
&= \frac{\partial}{\partial w_{ij}^1}\left(\frac{\partial\left(\frac{1}{2}\sum_{k=1}^{K}[d_k - z_k]^2\right)}{\partial w_{ij}^1}\right) \\
&= \frac{\partial}{\partial w_{ij}^1}\left(\frac{1}{2}\sum_{k=1}^{K}\frac{\partial([d_k - z_k]^2)}{\partial w_{ij}^1}\right) \\
&= \frac{\partial}{\partial w_{ij}^1}\left(\sum_{k=1}^{K}[d_k - z_k]\frac{\partial(-z_k)}{\partial w_{ij}^1}\right) \\
&= -\frac{\partial}{\partial w_{ij}^1}\left(\sum_{k=1}^{K}[d_k - z_k]\frac{\partial z_k}{\partial w_{ij}^1}\right) \\
&= \sum_{k=1}^{K}\frac{\partial z_k}{\partial w_{ij}^1}\frac{\partial z_k}{\partial w_{ij}^1} - \sum_{k=1}^{K}[d_k - z_k]\frac{\partial^2 z_k}{(\partial w_{ij}^1)^2}
\end{aligned}$$

Using the Levenberg and Marquardt approximation, the second term is assumed to be negligible [39], [50:523]. Essentially, this assumption is good when the second term is zero (as in the linear case), or when the second term is small or negligible compared to the term involving the first derivative [50:523]. Also the term multiplying the second derivative in the second term is $[d_k - z_k]$, which should just be the random measurement error of each point for a trained network. This error should fluctuate randomly around zero and should in general be uncorrelated with the network model, so the second derivative terms would tend to cancel out when summed over entire data set [50:523]. When the second term is negligible, $h_{jj}$ is guaranteed to be positive, and can be approximated as

$$h_{jj} = \sum_{k=1}^{K} \left(\frac{\partial z_k}{\partial w^1_{ij}}\right)^2$$

Substituting $h_{jj}$ into Equation 75 gives

$$d\mathcal{E}_o = \frac{1}{2}\sum_{j=1}^{H}\sum_{k=1}^{K}\left(\frac{\partial z_k}{\partial w^1_{ij}}\right)^2 (d\,w^1_{ij})^2, \tag{76}$$

where

$$\frac{\partial z_k}{\partial w^1_{ij}} = \delta^2_k \delta^1_j w^2_{jk} x_i,$$

and $\delta^2_k$ and $\delta^1_j$ are defined in Chapter 1, Section 3. Also, $d\,w^1_{ij}$ when the weight parameter is eliminated is $-w^1_{ij}$. Making these substitutions, the detailed neural network notation for Equation 76 becomes

$$d\mathcal{E}_o = \frac{1}{2}\sum_{j=1}^{H}\sum_{k=1}^{K}\left(-\delta^2_k \delta^1_j w^2_{jk} x_i w^1_{ij}\right)^2$$

which is the definition of error corresponding to the saliency $s_i(\mathbf{x}^p)$ of feature $i$ for the $p$th input vector, i.e.

$$s_i(\mathbf{x}^p) = \frac{1}{2}\sum_{j=1}^{H}\sum_{k=1}^{K}\left(\delta^2_k \delta^1_j w^2_{jk} x^p_i w^1_{ij}\right)^2$$

The second order saliency metric, $\bar{s}_i$ is formed by averaging $s_i(\mathbf{x})$ over all $P$ data vectors:

$$\bar{s}_i = P^{-1}\sum_{p=1}^{P} s_i(\mathbf{x}^p) \tag{77}$$

### *A.2 Relevance Feature Evaluation Metric*

Mozer and Smolensky propose a saliency technique for measuring the relevance of a neural network feature. The relevance of a neural network feature $i$, $\rho_i$, is measured as a function of how well the network performs with the feature versus how well the network performs without the feature.

$$\rho_i = \mathcal{E}_{\text{without feature } i} - \mathcal{E}_{\text{with feature } i}$$

where neural network error $\mathcal{E}$ is defined as the mean absolute error over all $P$ vectors: $\mathcal{E} = P^{-1}\sum_{p=1}^{P}\sum_{k=1}^{K}|d_k - z_k(\mathbf{x}^p, \hat{\mathbf{w}})|$.

A relevance factor $\alpha_i$ is introduced which corresponds to the attentional strength of feature $i$, $x_i$. Mozer and Smolensky's relevance factor $\alpha_i$ is associated with the feature unit $x_i$. Essentially, Mozer and Smolensky consider two discrete levels of relevance: $\alpha_i = 0$ and $\alpha_i = 1$, corresponding to $\mathcal{E}_{\text{without feature } i}$ and $\mathcal{E}_{\text{with feature } i}$ respectively. The neural network output $z_k$ can be defined as:

$$z_k = f\left(\sum_{j=0}^{H} x_j^1 w_{jk}^2\right),$$

where $\alpha_i$ is associated with feature $x_i$ as follows

$$x_j^1 = f\left(\sum_{i=1}^{M+1} x_i \alpha_i w_{ij}^1\right)$$

Mozer and Smolensky propose approximating $\rho_i$ using the derivative of the error with respect to $\alpha_i$:

$$\lim_{\gamma\to 1}\frac{\mathcal{E}_{\alpha_i=\gamma} - \mathcal{E}_{\alpha_i=1}}{\gamma - 1} = \left.\frac{\partial\mathcal{E}}{\partial\alpha_i}\right|_{\alpha_i=1}$$

and they assume this relationship holds approximately for $\gamma = 0$, giving

$$\hat{\rho}_i = -\frac{\partial\mathcal{E}}{\partial\alpha_i},$$

Using this definition of neural network error, the relevance of feature $i$, $\hat{\rho}_i$, can be derived in detailed neural network notation as follows

$$\begin{aligned}
\hat{\rho}_i &= \frac{\partial \mathcal{E}}{\partial \alpha_i} \\
&= \frac{\partial \left( P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} |d_k - z_k| \right)}{\partial \alpha_i} \\
&= P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} \frac{\partial |z_k|}{\partial \alpha_i} \\
&= P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} \left| \delta_k^2 \sum_{j=1}^{H} w_{jk}^2 \; \frac{\partial \left( x_j^1 \right)}{\partial \alpha_i} \right| \\
&= P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} \left| \delta_k^2 \sum_{j=1}^{H} \delta_j^1 w_{jk}^2 w_{ij}^1 x_i^p \right|
\end{aligned}$$

The relevance or saliency $\rho_i$ is essentially a weighted average of the terms which comprise $\hat{\Lambda}_i^{\text{data}}$ which is defined here for completeness.

$$\hat{\Lambda}_i^{\text{data}} = P^{-1} \sum_{p=1}^{P} \sum_{k=1}^{K} \left| \delta_k^2 \sum_{j=1}^{H} \delta_j^1 w_{jk}^2 w_{ij}^1 \right|$$

If a relevance factor $\alpha$ is defined which is associated with the weight parameters connected to a feature, rather than the feature itself, the resulting relevance of feature $i$ would be:

$$\hat{\rho}_i = P^{-1} \sum_{p=1}^{P} \sum_{j=1}^{H} \sum_{k=1}^{K} \left| \delta_k^2 \delta_j^1 w_{jk}^2 w_{ij}^1 x_i \right|$$

which is very similar $\bar{s}_i$ defined in Equation 77 in Sectionsec:dalone.

# Appendix B. *Model Selection Audit Runs*

## *B.1 Application of the Architecture Selection on the API Problem*

```
OVERALL NEURAL NETWORK SELECTION: AUDIT TRAIL
.........................................


Number of ANN iterations to find good local min
in architecture and feature modules   10
Number of ANN iterations for saliency screening module statistics   10
Initial model structure is:
NUMBER OF INITIAL MIDDLE NODES   8
NUMBER OF INITIAL FEATURES   4
NUMBER OF OUTPUTS   1
  1 OUTPUTS USED FOR   2 CLASSES
NUMBER OF VECTORS; TRAIN - TEST:  181  100
NUMBER OF INITIAL TRAINING EPOCHS   500
NUMBER OF ADDITIONAL TRAINING EPOCHS BETWEEN TEST SET SSE EVALUATIONS (m)  50
IMPROVEMENT REQUIRED BETWEEN TEST SET EVALUATIONS  FOR TRAINING TO CONTINUE (1-C3)% = 5.00000E-03%
TRAINING CONTINUES IF:ts-sse[t] ; TS-SSE[t-m]*C3
LOG declining learning rates used
MOMENTUM STEP SIZE  C2 =    0.300000
ONLY THE STATISTICALLY INDICATED FEATURES ARE REMOVED
STATISTICAL SIGNIF LEVELS: ARCH,FEAT,SAL=    5.0000000000000D-02    5.0000000000000D-02
    5.0000000000000D-02
RANDOM NUMBER SEED     10.00000

//////////////////////////////////////////////
/////////////////////////////////////////

************INITIAL ARCHITECTURE SELECTION MODULE:*********************

MODEL SELECTION ITERATION   1
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  8  4
CURRENT FEATURE SELECTION VECTOR
  1  1  1  1

SUMMARY OF   10 RUNS

  1 #eps=  550 FULL %ERR=     2.84698 % TOTAL SSE=    7.34563
  2 #eps=  700 FULL %ERR=     3.91459 % TOTAL SSE=    7.40277
  3 #eps=  550 FULL %ERR=     4.98221 % TOTAL SSE=    9.91086
  4 #eps=  600 FULL %ERR=     4.62633 % TOTAL SSE=   10.49259
  5 #eps=  550 FULL %ERR=     2.84698 % TOTAL SSE=    7.90157
  6 #eps=  550 FULL %ERR=     3.91459 % TOTAL SSE=    9.52083
  7 #eps=  550 FULL %ERR=     4.27046 % TOTAL SSE=    7.88130
  8 #eps=  550 FULL %ERR=     6.04982 % TOTAL SSE=    11.7386
  9 #eps=  600 FULL %ERR=     5.69395 % TOTAL SSE=    12.5137
  10 #eps=  650 FULL %ERR=     3.55872 % TOTAL SSE=    9.43272

SUMMARY OF   10 RUNS

  1 #eps=  550 REDU %ERR=     4.62633 % TOTAL SSE=    11.0259
  2 #eps=  550 REDU %ERR=     5.69395 % TOTAL SSE=    11.9587
  3 #eps=  650 REDU %ERR=     4.62633 % TOTAL SSE=    9.91830
  4 #eps=  850 REDU %ERR=     4.62633 % TOTAL SSE=    9.35481
```

```
 5 #eps=  550 RBDU %BRR=    2.84698 % TOTAL SSB=   7.17378
 6 #eps=  550 RBDU %BRR=    4.27046 % TOTAL SSB=   10.8573
 7 #eps=  550 RBDU %BRR=    4.27046 % TOTAL SSB=   9.06195
 8 #eps=  650 RBDU %BRR=    4.27046 % TOTAL SSB=   7.92945
 9 #eps=  650 RBDU %BRR=    3.91459 % TOTAL SSB=  10.02436
 10 #eps=  650 RBDU %BRR=    4.27046 % TOTAL SSB=   8.94219

Current feature select vector  1  1  1  1
Degrees of freedom  6  232
Full model minimum TOTAL SSB=   7.3456332677325
Reduced model minimum TOTAL SSB=   7.1737830674871
LIKBLIHOOD RATIO TBST STATISTIC L=  -0.90460198151696
Accept Reduced Model: SSB'R ¡ SSB'F  L ¡ F'ALPHA
RBDUCBD MODBL BBCOMBS FULL MODBL
APPROPRIATB NUMBBR OF MIDDLB NODBS  7

INVBSTIGATING RBDUCBD ARCHITBCTURB NBXT

////////////////////////////////////////
/////////////////////////////////////////

MODBL SBLBCTION ITBRATION  1
CURRBNT NUMBBR MIDDLB NODBS/FBATURBS - FULL MODBL  7  4
CURRBNT FBATURB SBLBCTION VBCTOR
 1  1  1  1
BBST FULL MODBL: TOTAL SSB=   7.1737830674871

SUMMARY OF   10 RUNS

 1 #eps=  550 RBDU %BRR=    4.98221 % TOTAL SSB=   11.6277
 2 #eps=  600 RBDU %BRR=    3.55872 % TOTAL SSB=   8.30053
 3 #eps=  550 RBDU %BRR=    4.62633 % TOTAL SSB=   11.7895
 4 #eps=  700 RBDU %BRR=    4.27046 % TOTAL SSB=   9.07790
 5 #eps=  750 RBDU %BRR=    3.20285 % TOTAL SSB=   7.99463
 6 #eps=  550 RBDU %BRR=    3.91459 % TOTAL SSB=   8.47713
 7 #eps=  550 RBDU %BRR=    3.20285 % TOTAL SSB=   8.63114
 8 #eps=  1000 RBDU %BRR=    2.49110 % TOTAL SSB=   4.36473
 9 #eps=  550 RBDU %BRR=    4.27046 % TOTAL SSB=   9.12368
 10 #eps=  550 RBDU %BRR=    2.84698 % TOTAL SSB=   7.01263

Current feature select vector  1  1  1  1
Degrees of freedom  6  238
Full model minimum TOTAL SSB=   7.1737830674871
Reduced model minimum TOTAL SSB=   4.3647347487185
LIKBLIHOOD RATIO TBST STATISTIC L=  -15.532332419718
Accept Reduced Model: SSB'R ¡ SSB'F   L ¡ F'ALPHA
RBDUCBD MODBL BBCOMBS FULL MODBL
APPROPRIATB NUMBBR OF MIDDLB NODBS  6

INVBSTIGATING RBDUCBD ARCHITBCTURB NBXT

////////////////////////////////////////
/////////////////////////////////////////

MODBL SBLBCTION ITBRATION  1
CURRBNT NUMBBR MIDDLB NODBS/FBATURBS - FULL MODBL  6  4
CURRBNT FBATURB SBLBCTION VBCTOR
 1  1  1  1
```

```
BEST FULL MODEL: TOTAL SSE=   4.3647347487185

SUMMARY OF   10 RUNS

 1 #eps=  550 REDU %ERR=    2.49110 % TOTAL SSE=    7.41936
 2 #eps=  850 REDU %ERR=    1.77936 % TOTAL SSE=    5.70470
 3 #eps=  550 REDU %ERR=    2.49110 % TOTAL SSE=    6.68141
 4 #eps=  600 REDU %ERR=    3.91459 % TOTAL SSE=    8.55718
 5 #eps=  700 REDU %ERR=    4.27046 % TOTAL SSE=    8.80496
 6 #eps=  550 REDU %ERR=    2.84698 % TOTAL SSE=    8.82592
 7 #eps=  650 REDU %ERR=    3.91459 % TOTAL SSE=    8.20973
 8 #eps=  550 REDU %ERR=    4.27046 % TOTAL SSE=   10.02347
 9 #eps=  1400 REDU %ERR=    2.13523 % TOTAL SSE=    6.49557
 10 #eps=  650 REDU %ERR=    4.27046 % TOTAL SSE=    11.5329

Current feature select vector   1  1  1  1
Degrees of freedom   6  244
Full model minimum TOTAL SSE=    4.3647347487185
Reduced model minimum TOTAL SSE=    5.7047028045141
LIKELIHOOD RATIO TEST STATISTIC L=    12.484615310248
Alpha1=    5.0000000000000D-02
REJECT REDUCED MODEL

APPROPRIATE NUMBER OF MIDDLE NODES  6

////////////////////////////////////////
/////////////////////////////////////////
FINAL NUMBER MIDDLE NODES/FEATURES  6  4
STOP=  1
PROGRAM COMPLETE
```

## B.2 Application of the Feature Selection Algorithm on the FLIR Problem

```
OVERALL NEURAL NETWORK SELECTION: AUDIT TRAIL
.......................................


Number of ANN iterations to find good local min
in architecture and feature modules   5
Number of ANN iterations for saliency screening module statistics   10
Initial model structure is:
NUMBER OF INITIAL MIDDLE NODES   8
NUMBER OF INITIAL FEATURES   9
NUMBER OF OUTPUTS   1
 1 OUTPUTS USED FOR   2 CLASSES
NUMBER OF VECTORS; TRAIN - TEST:   350  200
NUMBER OF INITIAL TRAINING EPOCHS   500
NUMBER OF ADDITIONAL TRAINING EPOCHS BETWEEN TEST SET SSE EVALUATIONS (m)  50
IMPROVEMENT REQUIRED BETWEEN TEST SET EVALUATIONS  FOR TRAINING TO CONTINUE
(1-C3)%  =  5.00000E-03%
TRAINING CONTINUES IF:ts-sse[t] ; TS-SSE[t-m]*C3
LOG declining learning rates used
MOMENTUM STEP SIZE  C2 =    0.300000
MAXIMUM NUMBER OF FINAL FEATURES TO BE SELECTED:  6
STATISTICAL SIGNIF LEVELS: ARCH,FEAT,SAL=    5.0000000000000D-02    5.0000000000000D-02
   5.0000000000000D-02
RANDOM NUMBER SEED      19.0000
```

```
/////////////////////////////////////////////
///////////////////////////////////////

************INITIAL ARCHITECTURE SELECTION MODULE:*********************

MODEL SELECTION ITERATION  1
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  8  9
CURRENT FEATURE SELECTION VECTOR
 1  1  1  1  1  1  1  1  1

SUMMARY OF  5 RUNS

 1 #eps=  650 FULL %ERR=    3.09091 % TOTAL SSE=   15.9061
 2 #eps=  550 FULL %ERR=    4.54545 % TOTAL SSE=   19.5941
 3 #eps=  600 FULL %ERR=    5.27273 % TOTAL SSE=   27.1451
 4 #eps=  550 FULL %ERR=    3.81818 % TOTAL SSE=   16.2628
 5 #eps=  600 FULL %ERR=    4.00000 % TOTAL SSE=   19.5451

SUMMARY OF  5 RUNS

 1 #eps=  600 REDU %ERR=    4.18182 % TOTAL SSE=   20.1310
 2 #eps=  650 REDU %ERR=    3.81818 % TOTAL SSE=   17.0385
 3 #eps=  550 REDU %ERR=    4.00000 % TOTAL SSE=   21.1545
 4 #eps=  600 REDU %ERR=    3.27273 % TOTAL SSE=   16.3270
 5 #eps=  550 REDU %ERR=    3.09091 % TOTAL SSE=   17.3828

Current feature select vector   1  1  1  1  1  1  1  1  1
Degrees of freedom   11  461
Full model minimum TOTAL SSE=    15.906069873657
Reduced model minimum TOTAL SSE=    16.326994202223
LIKELIHOOD RATIO TEST STATISTIC L=    1.1090455462504
Alpha1=    5.0000000000000D-02
Accept Reduced Model: L ; F'alpha
REDUCED MODEL BECOMES FULL MODEL
APPROPRIATE NUMBER OF MIDDLE NODES  7

INVESTIGATING REDUCED ARCHITECTURE NEXT

//////////////////////////////////////////
///////////////////////////////////////////

MODEL SELECTION ITERATION  1
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  7  9
CURRENT FEATURE SELECTION VECTOR
 1  1  1  1  1  1  1  1  1
BEST FULL MODEL: TOTAL SSE=    16.326994202223

SUMMARY OF  5 RUNS

 1 #eps=  600 REDU %ERR=    3.45455 % TOTAL SSE=   16.1392
 2 #eps=  550 REDU %ERR=    4.00000 % TOTAL SSE=   17.5262
 3 #eps=  550 REDU %ERR=    4.90909 % TOTAL SSE=   22.6289
 4 #eps=  600 REDU %ERR=    3.81818 % TOTAL SSE=   18.6528
 5 #eps=  600 REDU %ERR=    3.81818 % TOTAL SSE=   18.4436

Current feature select vector   1  1  1  1  1  1  1  1  1
Degrees of freedom   11  472
```

```
Full model minimum TOTAL SSE=    16.326994202223
Reduced model minimum TOTAL SSE=    16.139161526359
LIKELIHOOD RATIO TEST STATISTIC L=  -0.49364440658808
Accept Reduced Model: SSE^R ¡ SSE^F   L ¡ F^ALPHA
REDUCED MODEL BECOMES FULL MODEL
APPROPRIATE NUMBER OF MIDDLE NODES  6

INVESTIGATING REDUCED ARCHITECTURE NEXT

////////////////////////////////////////
/////////////////////////////////////////

MODEL SELECTION ITERATION  1
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  6  9
CURRENT FEATURE SELECTION VECTOR
 1  1  1  1  1  1  1  1  1
BEST FULL MODEL: TOTAL SSE=    16.139161526359
        .
SUMMARY OF   5 RUNS

  1 #eps=  600 REDU %ERR=     4.54545 % TOTAL SSE=    20.2141
  2 #eps=  600 REDU %ERR=     4.90909 % TOTAL SSE=    22.6786
  3 #eps=  550 REDU %ERR=     4.36364 % TOTAL SSE=    19.7243
  4 #eps=  600 REDU %ERR=     4.36364 % TOTAL SSE=    19.9564
  5 #eps=  550 REDU %ERR=     4.90909 % TOTAL SSE=    21.9964

Current feature select vector   1  1  1  1  1  1  1  1  1
Degrees of freedom   11   483
Full model minimum TOTAL SSE=    16.139161526359
Reduced model minimum TOTAL SSE=    19.724257955158
LIKELIHOOD RATIO TEST STATISTIC L=    9.7538106148135
Alpha1=    5.0000000000000D-02
REJECT REDUCED MODEL

APPROPRIATE NUMBER OF MIDDLE NODES  6

////////////////////////////////////////
/////////////////////////////////////////


REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
 0  1  1  1  1  1  1  1  1

SUMMARY OF   5 RUNS

  1 #eps=  550 REDU %ERR=     5.27273 % TOTAL SSE=    24.6785
  2 #eps=  600 REDU %ERR=     5.27273 % TOTAL SSE=    26.1357
  3 #eps=  650 REDU %ERR=     5.81818 % TOTAL SSE=    25.2757
  4 #eps=  600 REDU %ERR=     6.00000 % TOTAL SSE=    24.1635
  5 #eps=  550 REDU %ERR=     4.90909 % TOTAL SSE=    25.6628

REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
 1  0  1  1  1  1  1  1  1

SUMMARY OF   5 RUNS
```

```
  1 #eps=  650 REDU %ERR=    5.09091 % TOTAL SSE=   23.1256
  2 #eps=  550 REDU %ERR=    5.27273 % TOTAL SSE=   22.3473
  3 #eps=  650 REDU %ERR=    5.63636 % TOTAL SSE=   24.2026
  4 #eps=  550 REDU %ERR=    5.27273 % TOTAL SSE=   25.1004
  5 #eps=  600 REDU %ERR=    6.18182 % TOTAL SSE=   23.5854

REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  0  1  1  1  1  1  1

SUMMARY OF   5 RUNS

  1 #eps=  550 REDU %ERR=    4.90909 % TOTAL SSE=   22.2598
  2 #eps=  550 REDU %ERR=    5.27273 % TOTAL SSE=   23.2684
  3 #eps=  550 REDU %ERR=    6.54545 % TOTAL SSE=   28.4469
  4 #eps=  550 REDU %ERR=    5.45455 % TOTAL SSE=   25.5317
  5 #eps=  550 REDU %ERR=    3.81818 % TOTAL SSE=   18.4095

REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  1  0  1  1  1  1  1

SUMMARY OF   5 RUNS

  1 #eps=  550 REDU %ERR=    7.09091 % TOTAL SSE=   26.9639
  2 #eps=  900 REDU %ERR=    4.72727 % TOTAL SSE=   22.1923
  3 #eps=  550 REDU %ERR=    3.63636 % TOTAL SSE=   16.8445
  4 #eps=  600 REDU %ERR=    6.72727 % TOTAL SSE=   25.3198
  5 #eps=  600 REDU %ERR=    4.72727 % TOTAL SSE=   21.2901

REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  1  1  0  1  1  1  1

SUMMARY OF   5 RUNS

  1 #eps=  550 REDU %ERR=    7.45455 % TOTAL SSE=   32.9257
  2 #eps=  550 REDU %ERR=    5.63636 % TOTAL SSE=   26.8087
  3 #eps=  550 REDU %ERR=    7.63636 % TOTAL SSE=   30.2583
  4 #eps=  600 REDU %ERR=    6.36364 % TOTAL SSE=   29.3370
  5 #eps=  700 REDU %ERR=    8.18182 % TOTAL SSE=   37.5262

REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  1  1  1  0  1  1  1

SUMMARY OF   5 RUNS

  1 #eps=  600 REDU %ERR=    6.00000 % TOTAL SSE=   29.1942
  2 #eps=  550 REDU %ERR=    4.00000 % TOTAL SSE=   21.4321
  3 #eps=  650 REDU %ERR=    4.72727 % TOTAL SSE=   20.4873
  4 #eps=  550 REDU %ERR=    4.90909 % TOTAL SSE=   21.0179
  5 #eps=  750 REDU %ERR=    4.18182 % TOTAL SSE=   20.2190

REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  1  1  1  1  0  1  1
```

```
SUMMARY OF  5 RUNS

 1 #eps=  550 REDU %ERR=    6.00000 % TOTAL SSE=   26.8945
 2 #eps=  650 REDU %ERR=    4.90909 % TOTAL SSE=   23.1940
 3 #eps=  550 REDU %ERR=    5.09091 % TOTAL SSE=   24.0323
 4 #eps=  550 REDU %ERR=    4.18182 % TOTAL SSE=   20.7260
 5 #eps=  550 REDU %ERR=    6.54545 % TOTAL SSE=   30.1556

REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
 1  1  1  1  1  1  1  0  1

SUMMARY OF  5 RUNS

 1 #eps=  550 REDU %ERR=    3.09091 % TOTAL SSE=   16.5537
 2 #eps=  550 REDU %ERR=    4.90909 % TOTAL SSE=   19.6359
 3 #eps=  600 REDU %ERR=    6.36364 % TOTAL SSE=   31.2188
 4 #eps=  650 REDU %ERR=    5.27273 % TOTAL SSE=   21.7756
 5 #eps=  850 REDU %ERR=    3.45455 % TOTAL SSE=   16.8168

REDUCED MODEL MIDDLE NODES/FEATURES  6  8
CANDIDATE FEATURE SELECTION VECTOR:
 1  1  1  1  1  1  1  1  0

SUMMARY OF  5 RUNS

 1 #eps=  550 REDU %ERR=    4.18182 % TOTAL SSE=   21.6605
 2 #eps=  550 REDU %ERR=    4.72727 % TOTAL SSE=   22.3330
 3 #eps=  550 REDU %ERR=    6.54545 % TOTAL SSE=   31.4054
 4 #eps=  750 REDU %ERR=    3.45455 % TOTAL SSE=   17.5292
 5 #eps=  700 REDU %ERR=    6.36364 % TOTAL SSE=   27.3871

FEATURE selection iteration #   1
Degrees of freedom   6  483
Full model minimum SSE=    16.139161526359
Reduced model minimum SSE=    16.553680887828
LIKELIHOOD RATIO TEST STATISTIC L=    2.0675676703377
Alpha2 desired=    5.0000000000000D-02
ACCEPT REDUCED MODEL: FNVAR ¡ CNVAR and  L ¡ F^alpha

///////////////////////////////////////////
////////////////////////////////////////////

STOP=  0

************STRUCTURE SUBMODULE NEXT: ******************ITERATION=  2

MODEL SELECTION ITERATION   2
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  6  8
CURRENT FEATURE SELECTION VECTOR
 1  1  1  1  1  1  1  0  1
BEST FULL MODEL: TOTAL SSE=    16.553680887828

SUMMARY OF  5 RUNS

 1 #eps=  650 REDU %ERR=    4.00000 % TOTAL SSE=   17.4293
 2 #eps=  600 REDU %ERR=    2.54545 % TOTAL SSE=   14.5525
 3 #eps=  600 REDU %ERR=    3.45455 % TOTAL SSE=   16.3739
 4 #eps=  550 REDU %ERR=    7.27273 % TOTAL SSE=    31.9498
 5 #eps=  600 REDU %ERR=    9.81818 % TOTAL SSE=    36.7903

Current feature select vector   1  1  1  1  1  1  1  0  1
Degrees of freedom   10   480
Full model minimum TOTAL SSE=    16.553680687828
Reduced model minimum TOTAL SSE=    14.552529586897
LIKELIHOOD RATIO TEST STATISTIC L=   -5.9114525209628
Accept Reduced Model: SSE^R ¡ SSE^F    L ¡ F^ALPHA
REDUCED MODEL BECOMES FULL MODEL
APPROPRIATE NUMBER OF MIDDLE NODES  5

//////////////////////////////////////////
///////////////////////////////////////////


************FEATURE SUBMODULE NEXT: ********************ITERATION=  2


REDUCED MODEL MIDDLE NODES/FEATURES  5  7
CANDIDATE FEATURE SELECTION VECTOR:
 0  1  1  1  1  1  1  0  1

SUMMARY OF   5 RUNS

 1 #eps=  550 REDU %ERR=    7.27273 % TOTAL SSE=    30.8956
 2 #eps=  600 REDU %ERR=    6.18182 % TOTAL SSE=    30.5071
 3 #eps=  550 REDU %ERR=    8.00000 % TOTAL SSE=    31.2504
 4 #eps=  600 REDU %ERR=    8.36364 % TOTAL SSE=    33.1243
 5 #eps=  550 REDU %ERR=    6.90909 % TOTAL SSE=    34.8414

REDUCED MODEL MIDDLE NODES/FEATURES  5  7
CANDIDATE FEATURE SELECTION VECTOR:
 1  0  1  1  1  1  1  0  1

SUMMARY OF   5 RUNS

 1 #eps=  600 REDU %ERR=    6.00000 % TOTAL SSE=    27.2922
 2 #eps=  650 REDU %ERR=    4.72727 % TOTAL SSE=    20.6952
 3 #eps=  700 REDU %ERR=    4.00000 % TOTAL SSE=    17.3677
 4 #eps=  650 REDU %ERR=    5.81818 % TOTAL SSE=    23.4969
 5 #eps=  650 REDU %ERR=    7.09091 % TOTAL SSE=    34.0911

REDUCED MODEL MIDDLE NODES/FEATURES  5  7
CANDIDATE FEATURE SELECTION VECTOR:
 1  1  0  1  1  1  1  0  1

SUMMARY OF   5 RUNS

 1 #eps=  550 REDU %ERR=    6.18182 % TOTAL SSE=    30.2873
 2 #eps=  600 REDU %ERR=    4.54545 % TOTAL SSE=    20.3390
 3 #eps=  550 REDU %ERR=    6.72727 % TOTAL SSE=    28.6888
 4 #eps=  600 REDU %ERR=    4.54545 % TOTAL SSE=    22.2996
 5 #eps=  550 REDU %ERR=    5.45455 % TOTAL SSE=    23.5299

REDUCED MODEL MIDDLE NODES/FEATURES  5  7
CANDIDATE FEATURE SELECTION VECTOR:
 1  1  1  0  1  1  1  0  1
```

```
SUMMARY OF   5 RUNS

  1 #eps=  600 REDU %ERR=    6.54545 % TOTAL SSE=   37.3590
  2 #eps=  650 REDU %ERR=    4.54545 % TOTAL SSE=   23.8984
  3 #eps=  750 REDU %ERR=    6.90909 % TOTAL SSE=   28.8756
  4 #eps=  650 REDU %ERR=    6.36364 % TOTAL SSE=   36.1223
  5 #eps=  850 REDU %ERR=    3.45455 % TOTAL SSE=   17.0709

REDUCED MODEL MIDDLE NODES/FEATURES  5  7
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  1  1  0  1  1  0  1

SUMMARY OF   5 RUNS

  1 #eps=  650 REDU %ERR=    7.27273 % TOTAL SSE=   32.2246
  2 #eps=  550 REDU %ERR=    8.72727 % TOTAL SSE=   37.4507
  3 #eps=  600 REDU %ERR=    8.00000 % TOTAL SSE=   30.5737
  4 #eps=  600 REDU %ERR=    6.90909 % TOTAL SSE=   30.9639
  5 #eps=  550 REDU %ERR=    6.36364 % TOTAL SSE=   32.0839

REDUCED MODEL MIDDLE NODES/FEATURES  5  7
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  1  1  1  0  1  0  1

SUMMARY OF   5 RUNS

  1 #eps=  650 REDU %ERR=    3.63636 % TOTAL SSE=   21.2184
  2 #eps=  550 REDU %ERR=    5.27273 % TOTAL SSE=   20.8525
  3 #eps=  700 REDU %ERR=    5.09091 % TOTAL SSE=   20.2395
  4 #eps=  550 REDU %ERR=    7.63636 % TOTAL SSE=   25.4701
  5 #eps=  550 REDU %ERR=    5.63636 % TOTAL SSE=   22.7575

REDUCED MODEL MIDDLE NODES/FEATURES  5  7
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  1  1  1  1  0  0  1

SUMMARY OF   5 RUNS

  1 #eps=  550 REDU %ERR=    4.54545 % TOTAL SSE=   22.7780
  2 #eps=  650 REDU %ERR=    4.90909 % TOTAL SSE=   21.9502
  3 #eps=  550 REDU %ERR=    6.18182 % TOTAL SSE=   28.5988
  4 #eps=  550 REDU %ERR=    7.63636 % TOTAL SSE=   35.5400
  5 #eps=  650 REDU %ERR=    5.09091 % TOTAL SSE=   24.2772

REDUCED MODEL MIDDLE NODES/FEATURES  5  7
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  1  1  1  1  1  0  0

SUMMARY OF   5 RUNS

  1 #eps=  950 REDU %ERR=    4.54545 % TOTAL SSE=   18.1854
  2 #eps=  800 REDU %ERR=    4.72727 % TOTAL SSE=   19.3559
  3 #eps=  550 REDU %ERR=    6.18182 % TOTAL SSE=   23.7416
  4 #eps=  750 REDU %ERR=    3.63636 % TOTAL SSE=   17.8803
  5 #eps=  550 REDU %ERR=    5.63636 % TOTAL SSE=   25.2310

FEATURE selection iteration #   2
```

```
Degrees of freedom   5  499
Full model minimum SSE=    14.552529586697
Reduced model minimum SSE=    17.070916928162
LIKELIHOOD RATIO TEST STATISTIC L=    17.270884429918
Alpha2 desired=    5.0000000000000D-02
ACCEPT REDUCED MODEL: FNVAR ; CNVAR

////////////////////////////////////////
/////////////////////////////////////////

STOP=  0

************STRUCTURE SUBMODULE NEXT: ********************ITERATION=  3

MODEL SELECTION ITERATION   3
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  5  7
CURRENT FEATURE SELECTION VECTOR
 1  1  1  0  1  1  1  0  1
BEST FULL MODEL: TOTAL SSE=    17.070916928162

SUMMARY OF   5 RUNS

 1 #eps=  550 REDU %ERR=     7.09091 % TOTAL SSE=    31.6753
 2 #eps=  600 REDU %ERR=     5.27273 % TOTAL SSE=    25.4150
 3 #eps=  750 REDU %ERR=     4.54545 % TOTAL SSE=    19.3078
 4 #eps=  1300 REDU %ERR=     2.18182 % TOTAL SSE=    12.0990
 5 #eps=  600 REDU %ERR=     3.81818 % TOTAL SSE=    17.4842

Current feature select vector   1  1  1  0  1  1  1  0  1
Degrees of freedom   9  504
Full model minimum TOTAL SSE=    17.070916928162
Reduced model minimum TOTAL SSE=    12.099028470063
LIKELIHOOD RATIO TEST STATISTIC L=   -16.309947194121
Accept Reduced Model: SSE'R ; SSE'F   L ; F'ALPHA
REDUCED MODEL BECOMES FULL MODEL
APPROPRIATE NUMBER OF MIDDLE NODES  4

////////////////////////////////////////
/////////////////////////////////////////


************FEATURE SUBMODULE NEXT: ********************ITERATION=  3


REDUCED MODEL MIDDLE NODES/FEATURES  4  6
CANDIDATE FEATURE SELECTION VECTOR:
 0  1  1  0  1  1  1  0  1

SUMMARY OF   5 RUNS

 1 #eps=  550 REDU %ERR=     6.90909 % TOTAL SSE=    32.8491
 2 #eps=  600 REDU %ERR=     7.09091 % TOTAL SSE=    32.3501
 3 #eps=  550 REDU %ERR=     8.72727 % TOTAL SSE=    35.3371
 4 #eps=  600 REDU %ERR=     9.45455 % TOTAL SSE=    42.9853
 5 #eps=  600 REDU %ERR=     6.00000 % TOTAL SSE=    31.5726

REDUCED MODEL MIDDLE NODES/FEATURES  4  6
CANDIDATE FEATURE SELECTION VECTOR:
 1 0 1 0 1 1 1 0 1

SUMMARY OF  5 RUNS

 1 #eps=  550 REDU %ERR=    8.90909 % TOTAL SSE=   39.6345
 2 #eps=  600 REDU %ERR=    5.09091 % TOTAL SSE=   23.4250
 3 #eps=  550 REDU %ERR=    5.27273 % TOTAL SSE=   21.8055
 4 #eps=  600 REDU %ERR=    10.7273 % TOTAL SSE=   41.0231
 5 #eps=  650 REDU %ERR=    8.90909 % TOTAL SSE=   42.6789

REDUCED MODEL MIDDLE NODES/FEATURES  4  6
CANDIDATE FEATURE SELECTION VECTOR:
 1 1 0 0 1 1 1 0 1

SUMMARY OF  5 RUNS

 1 #eps=  550 REDU %ERR=    8.54545 % TOTAL SSE=   38.0540
 2 #eps=  550 REDU %ERR=    8.72727 % TOTAL SSE=   34.5625
 3 #eps=  550 REDU %ERR=    8.18182 % TOTAL SSE=   35.0380
 4 #eps=  550 REDU %ERR=    4.90909 % TOTAL SSE=   20.9889
 5 #eps=  600 REDU %ERR=    4.72727 % TOTAL SSE=   20.9025

REDUCED MODEL MIDDLE NODES/FEATURES  4  6
CANDIDATE FEATURE SELECTION VECTOR:
 1 1 1 0 0 1 1 0 1

SUMMARY OF  5 RUNS

 1 #eps=  750 REDU %ERR=    7.09091 % TOTAL SSE=   30.0034
 2 #eps=  550 REDU %ERR=    8.54545 % TOTAL SSE=   34.3897
 3 #eps=  550 REDU %ERR=   10.00000 % TOTAL SSE=   37.1891
 4 #eps=  550 REDU %ERR=    7.09091 % TOTAL SSE=   33.6600
 5 #eps=  600 REDU %ERR=    8.54545 % TOTAL SSE=   36.9951

REDUCED MODEL MIDDLE NODES/FEATURES  4  6
CANDIDATE FEATURE SELECTION VECTOR:
 1 1 1 0 1 0 1 0 1

SUMMARY OF  5 RUNS

 1 #eps=  600 REDU %ERR=    5.09091 % TOTAL SSE=   23.4959
 2 #eps=  550 REDU %ERR=    9.81818 % TOTAL SSE=   35.1912
 3 #eps=  550 REDU %ERR=    8.72727 % TOTAL SSE=   38.0612
 4 #eps=  600 REDU %ERR=    5.09091 % TOTAL SSE=   23.5077
 5 #eps=  650 REDU %ERR=    6.18182 % TOTAL SSE=   23.9230

REDUCED MODEL MIDDLE NODES/FEATURES  4  6
CANDIDATE FEATURE SELECTION VECTOR:
 1 1 1 0 1 1 0 0 1

SUMMARY OF  5 RUNS

 1 #eps=  550 REDU %ERR=    5.63636 % TOTAL SSE=   23.2047
 2 #eps=  550 REDU %ERR=    6.18182 % TOTAL SSE=   29.5208
 3 #eps=  550 REDU %ERR=    6.36364 % TOTAL SSE=   24.9051
 4 #eps=  550 REDU %ERR=    8.72727 % TOTAL SSE=   40.7081
 5 #eps=  650 REDU %ERR=    5.27273 % TOTAL SSE=   22.6001
```

```
REDUCED MODEL MIDDLE NODES/FEATURES  4  6
CANDIDATE FEATURE SELECTION VECTOR:
 1  1  1  0  1  1  1  0  0

SUMMARY OF   5 RUNS

 1 #eps=  600 REDU %ERR=    4.18182 % TOTAL SSE=    17.8449
 2 #eps=  600 REDU %ERR=    5.27273 % TOTAL SSE=    26.8046
 3 #eps=  550 REDU %ERR=    7.81818 % TOTAL SSE=    34.3018
 4 #eps=  600 REDU %ERR=    4.54545 % TOTAL SSE=    19.6977
 5 #eps=  550 REDU %ERR=    5.45455 % TOTAL SSE=    25.7884

FEATURE selection iteration #   3
Degrees of freedom   4  513
Full model minimum SSE=    12.099028470063
Reduced model minimum SSE=    17.844950420117
LIKELIHOOD RATIO TEST STATISTIC L=    60.906914296286
Alpha2 desired=    5.0000000000000D-02
ACCEPT REDUCED MODEL: FNVAR ; CNVAR
FINAL NUMBER OF MIDDLE NODES/FEATURES:    4/  6
FINAL FEATURE SELECTION VECTOR
 1  1  1  0  1  1  1  0  0


/////////////////////////////////////////
//////////////////////////////////////////

STOP=  0

************STRUCTURE SUBMODULE NEXT: ******************ITERATION=  4

MODEL SELECTION ITERATION   4
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  4  6
CURRENT FEATURE SELECTION VECTOR
 1  1  1  0  1  1  1  0  0
BEST FULL MODEL: TOTAL SSE=    17.844950420117

SUMMARY OF   5 RUNS

 1 #eps=  600 REDU %ERR=    4.90909 % TOTAL SSE=    22.1010
 2 #eps=  550 REDU %ERR=    5.27273 % TOTAL SSE=    22.2857
 3 #eps=  600 REDU %ERR=    4.00000 % TOTAL SSE=    19.4946
 4 #eps=  650 REDU %ERR=    7.63636 % TOTAL SSE=    31.1421
 5 #eps=  550 REDU %ERR=    6.36364 % TOTAL SSE=    27.3371

Current feature select vector   1  1  1  0  1  1  1  0  0
Degrees of freedom   8  517
Full model minimum TOTAL SSE=    17.844950420117
Reduced model minimum TOTAL SSE=    19.494557512129
LIKELIHOOD RATIO TEST STATISTIC L=    5.9740069774096
Alpha1=    5.0000000000000D-02
REJECT REDUCED MODEL

APPROPRIATE NUMBER OF MIDDLE NODES  4

/////////////////////////////////////////
//////////////////////////////////////////
```

```
FINAL NUMBER MIDDLE NODES/FEATURES:   4  6
FINAL FEATURE SELECTION VECTOR
 1  1  1  0  1  1  1  0  0


////////////////////////////////////////
/////////////////////////////////////////
PROGRAM COMPLETE
```

## B.3 Application of the Comprehensive Neural Network Selection Methodology on a Modified XOR Problem

```
OVERALL NEURAL NETWORK SELECTION: AUDIT TRAIL
.......................................


Number of ANN iterations to find good local min
in architecture and feature modules   5
Number of ANN iterations for saliency screening module statistics   10
Initial model structure is:
NUMBER OF INITIAL MIDDLE NODES   6
NUMBER OF INITIAL FEATURES   5
NUMBER OF OUTPUTS   1
  1 OUTPUTS USED FOR   2 CLASSES
NUMBER OF VECTORS; TRAIN - TEST:   300  200
NUMBER OF INITIAL TRAINING EPOCHS   500
NUMBER OF ADDITIONAL TRAINING EPOCHS BETWEEN TEST SET SSE EVALUATIONS (m)  50
IMPROVEMENT REQUIRED BETWEEN EVALUATIONS  FOR TRAINING TO CONTINUE (1-C3)%
   5.00000E-03%
TRAINING CONTINUES IF:ts-sse[t] ; TS-SSE[t-m]*C3
LOG declining learning rates used
MOMENTUM STEP SIZE  C2 =    0.300000
ONLY THE STATISTICALLY INDICATED FEATURES ARE REMOVED
STATISTICAL SIGNIF LEVELS: ARCH,FEAT,SAL=     . 000000000000D-02    5.0000000000000D-02
   5.0000000000000D-02
RANDOM NUMBER SEED      14.0000

////////////////////////////////////////////
///////////////////////////////////////

************INITIAL ARCHITECTURE SELECTION MODULE:**********************

MODEL SELECTION ITERATION   1
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL   6  5
CURRENT FEATURE SELECTION VECTOR
  1  1  1  1  1

SUMMARY OF   5 RUNS

  1 #eps=  550 FULL %ERR=    1.60000 % TOTAL SSE=    7.91194
  2 #eps=  550 FULL %ERR=    3.20000 % TOTAL SSE=    10.7792
  3 #eps=  700 FULL %ERR=    3.60000 % TOTAL SSE=    16.2286
  4 #eps=  600 FULL %ERR=    2.60000 % TOTAL SSE=   10.32850
  5 #eps=  550 FULL %ERR=    2.20000 % TOTAL SSE=   10.07596

SUMMARY OF   5 RUNS
```

```
 1 #eps=  600 REDU %ERR=    3.80000 % TOTAL SSE=   16.6088
 2 #eps=  650 REDU %ERR=    1.40000 % TOTAL SSE=   8.09073
 3 #eps=  550 REDU %ERR=    1.80000 % TOTAL SSE=  10.46256
 4 #eps=  550 REDU %ERR=    2.00000 % TOTAL SSE=   9.26759
 5 #eps=  550 REDU %ERR=    4.40000 % TOTAL SSE=   16.5097

Current feature select vector  1  1  1  1  1
Degrees of freedom  7  457
Full model minimum TOTAL SSE=   7.9119365043529
Reduced model minimum TOTAL SSE=   8.0907288547737
LIKELIHOOD RATIO TEST STATISTIC L=   1.4753134456558
Alpha1=   5.0000000000000D-02
Accept Reduced Model: L ¡ F^ALPHA
REDUCED MODEL BECOMES FULL MODEL
APPROPRIATE NUMBER OF MIDDLE NODES  5

INVESTIGATING REDUCED ARCHITECTURE NEXT

/////////////////////////////////////////
//////////////////////////////////////////

MODEL SELECTION ITERATION  1
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  5  5
CURRENT FEATURE SELECTION VECTOR
 1  1  1  1  1
BEST FULL MODEL: TOTAL SSE=   8.0907288547737

SUMMARY OF  5 RUNS

 1 #eps=  600 REDU %ERR=    2.40000 % TOTAL SSE=   10.6701
 2 #eps=  550 REDU %ERR=    1.80000 % TOTAL SSE=   9.59482
 3 #eps=  650 REDU %ERR=    1.40000 % TOTAL SSE=   8.33807
 4 #eps=  550 REDU %ERR=    3.20000 % TOTAL SSE=   14.3301
 5 #eps=  750 REDU %ERR=    1.60000 % TOTAL SSE=   8.37509

Current feature select vector  1  1  1  1  1
Degrees of freedom  7  464
Full model minimum TOTAL SSE=   8.0907288547737
Reduced model minimum TOTAL SSE=   8.3380696448074
LIKELIHOOD RATIO TEST STATISTIC L=   2.0264133471364
Alpha1=   5.0000000000000D-02
REJECT REDUCED MODEL

APPROPRIATE NUMBER OF MIDDLE NODES  5

/////////////////////////////////////////
//////////////////////////////////////////


************ SALIENCY SCREENING MODULE:*********************


SUMMARY OF  10 TRAINING RUNS

 1 #eps=  550 REDU %ERR=    2.40000 % TOTAL SSE=  10.26702
 2 #eps=  700 REDU %ERR=    1.60000 % TOTAL SSE=   7.46934
 3 #eps=  550 REDU %ERR=    1.60000 % TOTAL SSE=   9.15596
 4 #eps=  550 REDU %ERR=     2.20000 % TOTAL SSE=    8.12369
 5 #eps=  600 REDU %ERR=     2.80000 % TOTAL SSE=    9.81324
 6 #eps=  700 REDU %ERR=     1.80000 % TOTAL SSE=    9.98154
 7 #eps=  600 REDU %ERR=     6.60000 % TOTAL SSE=    24.8585
 8 #eps=  750 REDU %ERR=     3.80000 % TOTAL SSE=    14.7960
 9 #eps=  600 REDU %ERR=     3.40000 % TOTAL SSE=    11.2866
 10 #eps=  650 REDU %ERR=     2.40000 % TOTAL SSE=    9.08073

NUMBER OF NOISE-LIKE FEATURES REMOVED WITH SALIENCY SCREENING   2
FEATURE SELECTION VECTOR AFTER SALIENCY SCREENING
 1  1  0  0  1

////////////////////////////////////////
/////////////////////////////////////////


************STRUCTURE SUBMODULE NEXT: ******************ITERATION=  1

MODEL SELECTION ITERATION   1
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  5  3
CURRENT FEATURE SELECTION VECTOR
 1  1  0  0  1

SUMMARY OF   5 RUNS

 1 #eps=  700 FULL %ERR=     1.80000 % TOTAL SSE=    7.64268
 2 #eps=  700 FULL %ERR=     1.80000 % TOTAL SSE=    8.00643
 3 #eps=  550 FULL %ERR=     1.60000 % TOTAL SSE=    8.06628
 4 #eps=  750 FULL %ERR=     2.00000 % TOTAL SSE=    8.76789
 5 #eps=  550 FULL %ERR=    0.400000 % TOTAL SSE=    7.82388

SUMMARY OF   5 RUNS

 1 #eps=  600 REDU %ERR=     17.8000 % TOTAL SSE=    60.9630
 2 #eps=  550 REDU %ERR=     2.60000 % TOTAL SSE=    12.5482
 3 #eps=  600 REDU %ERR=     1.40000 % TOTAL SSE=    7.65013
 4 #eps=  700 REDU %ERR=     1.60000 % TOTAL SSE=    7.38523
 5 #eps=  600 REDU %ERR=     1.80000 % TOTAL SSE=    7.58410

Current feature select vector   1  1  0  0  1
Degrees of freedom   5  464
Full model minimum TOTAL SSE=    7.6426826397289
Reduced model minimum TOTAL SSE=    7.3852344052716
LIKELIHOOD RATIO TEST STATISTIC L=   -3.1260222730495
Accept Reduced Model: SSE'R ¡ SSE'F
REDUCED MODEL BECOMES FULL MODEL
APPROPRIATE NUMBER OF MIDDLE NODES  4

INVESTIGATING REDUCED ARCHITECTURE

////////////////////////////////////////
/////////////////////////////////////////

MODEL SELECTION ITERATION   1
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  4  3
CURRENT FEATURE SELECTION VECTOR
 1  1  0  0  1
BEST FULL MODEL: TOTAL SSE=    7.3852344052716
```

```
SUMMARY OF   5 RUNS

  1 #eps=  550 REDU %ERR=     5.00000 % TOTAL SSE=    16.2218
  2 #eps=  550 REDU %ERR=     18.4000 % TOTAL SSE=    62.6749
  3 #eps=  650 REDU %ERR=     4.00000 % TOTAL SSE=    17.5724
  4 #eps=  600 REDU %ERR=     18.2000 % TOTAL SSE=    60.4270
  5 #eps=  600 REDU %ERR=     4.00000 % TOTAL SSE=    16.2336

Current feature select vector   1  1  0  0  1
Degrees of freedom   5  469
Full model minimum TOTAL SSE=    7.3852344052716
Reduced model minimum TOTAL SSE=    16.221757126865
LIKELIHOOD RATIO TEST STATISTIC L=    112.23283998864
Alpha1=    5.0000000000000D-02
REJECT REDUCED MODEL

APPROPRIATE NUMBER OF MIDDLE NODES  4

//////////////////////////////////////////
///////////////////////////////////////////

************FEATURE SUBMODULE NEXT: *******************ITERATION=  1


REDUCED MODEL MIDDLE NODES/FEATURES  4  2
CANDIDATE FEATURE SELECTION VECTOR:
  0  1  0  0  1

SUMMARY OF   5 RUNS

  1 #eps=  1250 REDU %ERR=     1.20000 % TOTAL SSE=    6.49672
  2 #eps=  600 REDU %ERR=     1.60000 % TOTAL SSE=    8.81672
  3 #eps=  550 REDU %ERR=     4.20000 % TOTAL SSE=    19.1158
  4 #eps=  750 REDU %ERR=     3.80000 % TOTAL SSE=    16.3031
  5 #eps=  850 REDU %ERR=     1.20000 % TOTAL SSE=    8.34364

REDUCED MODEL MIDDLE NODES/FEATURES  4  2
CANDIDATE FEATURE SELECTION VECTOR:
  1  0  0  0  1

SUMMARY OF   5 RUNS

  1 #eps=  550 REDU %ERR=     51.8000 % TOTAL SSE=    124.733
  2 #eps=  550 REDU %ERR=     51.8000 % TOTAL SSE=    125.079
  3 #eps=  550 REDU %ERR=     49.6000 % TOTAL SSE=    124.842
  4 #eps=  550 REDU %ERR=     46.2000 % TOTAL SSE=    125.595
  5 #eps=  550 REDU %ERR=     48.2000 % TOTAL SSE=    124.645

REDUCED MODEL MIDDLE NODES/FEATURES  4  2
CANDIDATE FEATURE SELECTION VECTOR:
  1  1  0  0  0

SUMMARY OF   5 RUNS

  1 #eps=  750 REDU %ERR=     3.20000 % TOTAL SSE=    14.6345
  2 #eps=  550 REDU %ERR=     17.8000 % TOTAL SSE=    60.4645
```

```
 3 #eps=  1250 REDU %ERR=     1.40000 % TOTAL SSE=    7.85197
 4 #eps=  600 REDU %ERR=     3.60000 % TOTAL SSE=    17.6820
 5 #eps=  1200 REDU %ERR=     2.20000 % TOTAL SSE=    10.9745

FEATURE selection iteration #   1
Degrees of freedom   4  469
Full model minimum SSE=    7.3852344052716
Reduced model minimum SSE=    6.4957242485871
LIKELIHOOD RATIO TEST STATISTIC L=   -14.122106374419
ACCEPT REDUCED MODEL: SSE^R ¡ SSE^F

/////////////////////////////////////////
//////////////////////////!///////////////

STOP=  0

************STRUCTURE SUBMODULE NEXT: *******************ITERATION=  2

MODEL SELECTION ITERATION   2
CURRENT NUMBER MIDDLE NODES/FEATURES - FULL MODEL  4  2
CURRENT FEATURE SELECTION VECTOR
 0  1  0  0  1
BEST FULL MODEL: TOTAL SSE=    6.4957242485871

SUMMARY OF   5 RUNS

 1 #eps=  600 REDU %ERR=     18.2000 % TOTAL SSE=    60.7789
 2 #eps=  550 REDU %ERR=     5.40000 % TOTAL SSE=    21.5857
 3 #eps=  600 REDU %ERR=     5.40000 % TOTAL SSE=    18.4563
 4 #eps=  550 REDU %ERR=     18.2000 % TOTAL SSE=    60.3443
 5 #eps=  600 REDU %ERR=     3.40000 % TOTAL SSE=    16.4279

Current feature select vector   0  1  0  0  1
Degrees of freedom   4  473
Full model minimum TOTAL SSE=    6.4957242485871
Reduced model minimum TOTAL SSE=    16.427880500077
LIKELIHOOD RATIO TEST STATISTIC L=    180.80777936258
Alpha1=    5.0000000000000D-02
REJECT REDUCED MODEL

APPROPRIATE NUMBER OF MIDDLE NODES  4

/////////////////////////////////////////
//////!///////////////////////////////////


************FEATU RE SUBMODULE NEXT: ********************ITERATION=  2


REDUCED MODEL MIDDLE NODES/FEATURES  4  1
CANDIDATE FEATURE SELECTION VECTOR:
 0  0  0  0  1

SUMMARY OF   5 RUNS

 1 #eps=  600 REDU %ERR=     50.4000 % TOTAL SSE=    124.963
 2 #eps=  600 REDU %ERR=     46.8000 % TOTAL SSE=    125.793
 3 #eps=  600 REDU %ERR=     50.2000 % TOTAL SSE=    124.773
 4 #eps=  600 REDU %ERR=    46.4000 % TOTAL SSE=    124.680
 5 #eps=  650 REDU %ERR=    48.2000 % TOTAL SSE=    124.749

REDUCED MODEL MIDDLE NODES/FEATURES  4  1
CANDIDATE FEATURE SELECTION VECTOR:
 0  1  0  0  0

SUMMARY OF   5 RUNS

 1 #eps=  550 REDU %ERR=    47.0000 % TOTAL SSE=    124.597
 2 #eps=  550 REDU %ERR=    51.8000 % TOTAL SSE=    124.548
 3 #eps=  550 REDU %ERR=    48.4000 % TOTAL SSE=    124.431
 4 #eps=  550 REDU %ERR=    46.2000 % TOTAL SSE=    124.718
 5 #eps=  550 REDU %ERR=    47.6000 % TOTAL SSE=    125.189

FEATURE selection iteration #   2
Degrees of freedom   4  473
Full model minimum SSE=    6.4957242485871
Reduced model minimum SSE=    124.43057468684
LIKELIHOOD RATIO TEST STATISTIC L=    2146.9193473471
Alpha2 desired=    5.0000000000000D-02
REJECT REDUCED MODEL
FINAL NUMBER OF MIDDLE NODES/FEATURES:    4/  1
FINAL FEATURE SELECTION VECTOR
 0  1  0  0  1


/////////////////////////////////////////////
//////////////////////////////////////////////

STOP=  1

************SALIENCY MODULE NEXT: ********************ITERATION=  3


SUMMARY OF   10 TRAINING RUNS

 1 #eps=  550 REDU %ERR=    2.00000 % TOTAL SSE=    7.57726
 2 #eps=  600 REDU %ERR=    1.40000 % TOTAL SSE=    8.56912
 3 #eps=  550 REDU %ERR=    4.40000 % TOTAL SSE=    16.9252
 4 #eps=  550 REDU %ERR=    1.60000 % TOTAL SSE=    10.8887
 5 #eps=  550 REDU %ERR=    4.40000 % TOTAL SSE=    17.3659
 6 #eps=  900 REDU %ERR=    1.40000 % TOTAL SSE=    7.73438
 7 #eps=  650 REDU %ERR=    2.20000 % TOTAL SSE=    9.11519
 8 #eps=  600 REDU %ERR=    2.00000 % TOTAL SSE=    9.48292
 9 #eps=  550 REDU %ERR=    5.40000 % TOTAL SSE=    19.1411
 10 #eps=  600 REDU %ERR=    4.00000 % TOTAL SSE=    15.5100

FINAL NUMBER MIDDLE NODES/FEATURES:    4  2
FINAL FEATURE SELECTION VECTOR
 0  1  0  0  1

/////////////////////////////////////////////
//////////////////////////////////////////////

PROGRAM COMPLETE
```

# *Bibliography*

1. Akaike, H. A New Look at the Statistical Model Identification, *IEEE Transactions on Automatic Control 19:6*, 716-723, December 1974.

2. Allen, D.M. Mean Square Error of Prediction as a Criterion for Selecting Variables, *Technometrics 13*, 469-475, 1971.

3. Anderson, T.W. *An Introduction to Multivariate Statistical Analysis*, John Wiley & Sons, New York, 1984.

4. Belue, L.M. *An Investigation of Multilayer Perceptrons for Classification*, Masters Thesis, Air Force Institute of Technology, March 1992.

5. Belue, L.M., and K.W. Bauer. Methods of Determining Input Features for Multilayer Perceptrons, Accepted for publication in *Neural Computing* in August 1993.

6. Belue, L.M., and K.W. Bauer. Determining Incendiary Functioning with Multilayer Perceptrons, Technical Paper, Wright Labs, Wright Patterson Air Force Base, forthcoming, 1994.

7. Belue, L.M., and K.W. Bauer. Methods of Determining Input Features for Multilayer Perceptrons, Working Paper Series 93, Department of Operational Sciences, School of Engineering, Air Force Institute of Technology, February 1993.

8. Ben-Basset, M. Use of Distance Measures, Information Measures and Error Bounds in Feature Evaluation, *Handbook of Statistics 2*, eds. P.R. Krishnaiah and L.N. Kanal, North Holland, 773-791, 1982.

9. Bichsel, M. and P. Seitz. Minimum Class Entropy: A Maximum Information Approach to Layered Networks, *Neural Networks 2*. 133-141, 1989.

10. Cover, T.M. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition, *IEEE Transactions on Electronic Computers*, EC-14:326-334, June 1965.

11. Cover, T.M. The Best Two Independent Measurements Are Not the Two Best. *IEEE Trans. Systems Man Cybernet. 4*, 116-117, 1974.

12. Cybenko, G. Approximation by Superpositions of a Sigmoidal Function, *Math. Control, Signals Sys. 2*, 303-314, 1989.

13. de la Maza, M. SPLITnet Dynamically Adjusting the Number of Hidden Units in a Neural Network, *Artificial Neural Networks*. eds. T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, North Holland, 647-651, 1991.

14. Devijver, P.A. and J. Kittler. *Pattern Recognition*. Prentice Hall, Englewood Cliffs, New Jersey, 1982.

15. Dillon, W.R. and M. Goldstein. *Multivariate Analysis Methods and Applications*, John Wiley & Sons, New York, 1984.

16. Elashoff, J.D., and others. On the Choice of Variables in Classification Problems with Dichotomous Variables, *Biometrika 54*, 668-670.

17. Foley, D.H. Considerations of Sample and Feature Sise, *IEEE Transactions on Information Theory, 18:* 618-626, September 1972.

18. Furnival, G.M. and R.W. Wilson, Jr. Regression by Leaps and Bounds, *Technometrics, 16:4*, 499-511.

19. Gallant A.R. *Nonlinear Statistical Models*, John Wiley & Sons, New York, 1987.

20. Giles, L.C. and T. Maxwell. Learning Invariance, and Generalization in High-Order Neural Networks, *Applied Optics, 26*: 4972-4978, December 1987.

21. Guo, Z. and R.E. Uhrig. Sensitivity Analysis and Applications to Nuclear Power Plant, *International Joint Conference on Neural Networks, II*, 453-457, June 7-11, 1992.

22. Hashem, S. Sensitivity Analysis for Feedforward Artificial Neural Networks with Differentiable Activation Functions, *International Joint Conference on Neural Networks, I*, 419-424, June 7-11, 1992.

23. Hecht-Nielsen, R. *Neurocomputing*, Addison-Wesley Publishing Company, New York, 1989.

24. Hines, W.W., and D.C. Montgomery. *Probability and Statistics in Engineering and Management Science, 2nd ed*, John Wiley & Sons, New York, 1980.

25. Hirose, Y., K. Yamashita, and S. Hijiya. Back-Propagation Algorithm Which Varies the Number of Hidden Units, *Neural Networks 4*. 61-66, 1991.

26. Hocking, R.R. The Analysis and Selection of Variables in Linear Regression, *Biometrics 32*, 1-49, March 1976.

27. Hornik, K., and others. Multilayer feedforward networks are universal approximators, *Neural Networks 2*, 359-368, 1989.

28. Huang, S. and Y. Huang. Bounds on the Number of Hidden Neurons in Multilayer Perceptrons, *IEEE Transactions on Neural Networks 2:1*, 47-55, Jan 1991.

29. James, M. *Classification Algorithms*, John Wiley & Sons, New York, 1985

30. Kanaya, F., and S. Miyake. Bayes Statistical Behavior and Valid Generalization of Pattern Classifying Neural Networks, *IEEE Transactions on Neural Networks. Vol. 2, No. 4*, 471-475, July 1991.

31. Karayiannis, N.B. ALADIN: Algorithms for Learning and Architecture DetermINation, *International Joint Conference on Neural Networks, IV*, 601-606, June 7-11, 1992.

32. Karnin, E.D. A Simple Procedure for Pruning Back-Propagation Trained Neural Networks, *IEEE Transactions on Neural Networks 1:2*,239-242, June 1990.

33. Klimasauskas, C.C. Neural Nets Tell Why, *Dr. Dobb's Journal*, April 1991.

34. Knight, Earl E. *Predicting Soviet Armor Piercing Incendiary Projectile Effects After Impact of Composite Materials*. MS Thesis, AFIT/GOR/ENS/92M. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 1992.

35. Krzanowski, W.J. *Principles of Multivariate Analysis A Users Perspective*, Clarendon Press, Oxford, 1988.

36. Krishnaiah, P.R. Selection of Variables Under Univariate Regression Models, *Handbook of Statistics 2*. eds. P.R. Krishnaiah and L.N. Kanal, North Holland, 805-820, 1982.

37. Kung, S.Y. and J.N. Hwang. An Algebraic Analysis for Optimal Hidden Units Size and Learning Rates in Back-Propagation Learning, *International Conference on Neural Networks I*, 363-370, San Diego, CA, July 24-27, 1988.

38. Lay, S.R. *Analysis with an Introduction to Proof*, Prentice Hall, New Jersey, 2nd edition, 1990.

39. Le Cun, Y., and others. Optimal Brain Damage. *Neural Information Processing Systems II*, 598-605, ed. D.S. Touretzky, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

40. Lippmann, R.P. An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, 4-22, April 1987.

41. Lippmann, R.P. Pattern Classification Using Neural Networks, *IEEE Communications Magazine 27*, 47-65, November 1989.

42. Mallows, C.L. Some Comments on *C* sub *p*, *Technometrics 15*, 661-675, 1973.

43. Mantel, N. Why Stepdown Procedures in Variable Selection, *Technometrics 12:2*, 621 625, August 1970.

44. Mitchell, J. A geometric Interpretation of Hidden Units in Feed Forward Neural Networks, *Network: Computation in Neural Systems 3*. 19-25, 1992.

45. Mozer, M. C. and P. Smolensky. Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment, *Advances in Neural Information Systems I*. Ed. David S. Touretzky, Morgan Kaufman Publishing Company, San Mateo, CA.

46. Myers, R.H. *Classical and Modern Regression with Applications, 2nd ed.*, PWS-KENT Publishing Company, Boston, 1990.

47. Neter, J., W. Wasserman, and M.H. Kutner. *Applied Linear Statistical Models Regression, 3rd ed.*, IRWIN, INC., Boston, 1990.

48. Parker, D. *Learning Logic*. Invention Report 581-64, File 1, Office of Technology Licensing, Stanford University, October 1982.

49. Pope, P.T. and J.T. Webster. The Use of an F-Statistic in Stepwise Regression Procedures, *Technometrics 14:2*, 327-340, May 1972.

50. Press, W.H., and others. *Numerical Recipes, The Art of Scientific Computing (Fortran Version)*, Cambridge University Press, 1989.

51. Priddy, K. L., and others. Bayesian Selection of Important Features for Feedforward Neural Networks. Accepted for publication in *Neurocomputing*: Paper Number: 92-92.

52. Redding, N.J., and others. Higher Order Separability and Minimual Hidden-Unit Fan-In, *Artificial Neural Networks*. eds. T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, North Holland, 25-30, 1991.

53. Reinhart, G.L. *A Fortran Based Learning System Using Multilayer Back-Propagation Neural Network Techniques*, Masters Thesis, Air Force Institute of Technology, March 1994.

54. Richard, M.D., and R.P. Lippmann, Neural Network Classifiers Estimate Bayesian *a posteriori* Probabilities, *Neural Computation 3*, 461-483, 1991.

55. Rogers, S.K., and others. *An Introduction to Biological and Artificial Neural Networks*, Air Force Institute of Technology, October, 1991.

56. Roggemann, M. C. *Multiple Sensor Fusion for Detecting Targets in FLIR and Range Images,* Phd Dissertation, Air Force Institute of Technology, December 1989.

57. Roggemann, M. C., and others. An Approach to Multiple Sensor Target Detection, *Proceedings of SPIE Conference on Sensor Fusion,* Volume 1100, 1989.

58. Roggemann, M. C., and others. Multiple Sensor Tactical Target Detection in FLIR and Range Images, in *Proceedings of Tri-Service Data Fusion Symposium,* May 1989.

59. Ruck, D.W., and others. Feature Selection Using a Multilayer Perceptron, *The Journal of Neural Network Computing, 2(2)*:40-48, Fall 1990.

60. Ruck, D.W. *Characterization of Multilayer Perceptrons and their Application to Multisensor Automatic Target Detection*, Phd Dissertation, Air Force Institute of Technology, December 1990.

61. Ruck, D.W., and others. The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function, *IEEE Transactions on Neural Networks. Vol. 1, No. 4*, 296-298, December 1990.

62. Ruck, D.W Personal Interview at Air Force Institute of Technology, 5 November 1993.

63. Rumelhart D., and others. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Chapter 8.* MIT Press, Cambridge, MA 1986.

64. Sartori, M.A. and P.J. Antsaklis. A Simple Method to Derive Bounds on the Size and to Train Multilayer Neural Networks, *IEEE Transactions on Neural Networks 2:4*, 467-471, July 1991.

65. Schmidhammer, J.L. On the Selection of Variables Under Regression Models Using Krishnaiah's Finite Intersection Tests, *Handbook of Statistics 2*, eds. P.R. Krishnaiah and L.N. Kanal, North Holland, 821-833, 1982.

66. Seber, G. A. F., and C. J. Wild. *Nonlinear Regression,* John Wiley & Sons, New York, 1989.

67. Spiegel, Murray R. *Schaum's Outline Series: Theory and Problems of Advanced Calculus,* McGraw-Hill Book Company, NY, 1963.

68. Shoemaker, P.A. A Note on Least-Squares Learning Procedures and Classification by Neural Network Models, *IEEE Transactions on Neural Networks. Vol. 2, No. 1,* 158-160, January 1991.

69. Sietsma J. and R.J.F. Dow. Neural Net Pruning- Why and How, *International Conference on Neural Networks I*, 325-333, San Diego, CA, July 24-27, 1988.

70. Steppe, J.M. *Speciality Exam in Operations Research,* Department of Operational Sciences, May 1992.

71. Schwarz, G. Estimating the Dimension of a Model, *The Annals of Statistics 6:2*, 461-464, 1978.

72. Tarr, G.L. *Multi-Layered Feedforward Neural Networks for Image Segmentation.* Phd Dissertation, Air Force Institute of Technology, December 1991.

73. Thompson, M.L. Selection of Variables in Multiple Regression Part 1: a Review and Evaluation, *International Statistical Review 46*, 1-19, 1978.

74. Van Campenhout, J.M. Topics in Measurement Selection, *Handbook of Statistics 2*, eds. P.R. Krishnaiah and L.N. Kanal, North Holland, 793-803, 1982.

75. Wan, E.A. Neural Network Classification: A Bayesian Interpretation, *IEEE Transactions on Neural Networks. Vol. 1, No. 4*, 303-305, December 1990.

76. Weiss, S.M. and C.A. Kulikowski. *Computer Systems that Learn*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

77. Werbos, P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD Dissertation, Harvard University, 1974.

78. Werbos, P. Applications of Advances in Nonlinear Sensitivity Analysis, *Systems Modeling and Optimization, proceedings of the 10th IFIP Conference, September 1981.* eds. R.F. Drenick and F. Kosin, Springer-Verlag, 1982.

79. White, H. An Additional Hidden Unit Test for Neglected Nonlinearity in Multilayer Feedforward Networks, *IEEE INNS International Joint Conference on Neural Networks II*, 451-455, Washington, DC, June 18-22, 1989.

80. White, H. *Artificial Neural Networks Approximation & Learning Theory,* Blackwell Publishers, Cambridge, USA, 1993.

81. White, H. Asymptotic Results for Learning in Single Hidden-Layer Feedforward Network Models, *Journal of the American Statistical Association* 84:408, December 1989.

82. White, H. Learning in Artificial Neural Networks: A Statistical Perspective, *Neural Computation 1*, 425-464, 1989.

83. Yang, X. A Convenient Method to Prune Multilayer Neural Networks via Transform Domain Backpropagation Algorithm, *International Joint Conference on Neural Networks, III*, 817-822, Baltimore, MD, June 7-11, 1992.

## *Vita*

Captain Jean Marie Steppe was born on April 7, 1963 in Bangor, Maine. She graduated from Beaver High School in Beaver, PA in 1981 and later attended the Air Force Academy for her undergraduate degree. In 1986, Captain Steppe graduated from the United States Air Force Academy with a Bachelor of Science in Operations Research. As a lieutenant, she served as a weapons and tactics analyst for the 4486 Fighter Weapons Squadron at Eglin AFB, Florida. She received her Master's of Science in Operations Research from the Air Force Institute of Technology (AFIT) at Wright-Patterson Air Force Base in 1991. Upon completion of her master's, she received a follow-on assignment at AFIT to pursue doctoral studies in Operations Research. Captain Steppe completed her Ph.D. and was assigned to the Air Mobility Command Headquarters at Scott Air Force Base, Illinois. She is married to Robert William Steppe and has three children: Holynne Marie, Patrick Neil, and Siobhan Marie.

Permanent address: Box 357
Beaver, PA 15009

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

| | | |
|---|---|---|
| 1. AGENCY USE ONLY (*Leave blank*) | 2. REPORT DATE<br>June 1994 | 3. REPORT TYPE AND DATES COVERED<br>Doctoral Dissertation |

4. TITLE AND SUBTITLE
Feature and Model Selection in Feedforward Neural Networks

5. FUNDING NUMBERS

6. AUTHOR(S)
Jean M. Steppe, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/DS/ENS/94J-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
WL/FIVS
Wright-Patterson AFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This research advances feature and model selection for feedforward neural networks. Feature selection involves determining a good feature subset given a set of candidate features. Model selection involves determining an appropriate architecture (number of middle nodes) for the neural network. Specific advances are made in: neural network feature saliency metrics used for evaluating or ranking features, statistical identification of irrelevant/noisy features, and statistical investigation of reduced neural network architectures and reduced feature subsets. New feature saliency metrics are presented which provide a more succinct quantitative measure of a feature's importance than other similar metrics. A catalogue of feature saliency metric definitions and interrelationships is also developed which consolidates the set of available metrics for the neural network practitioner. The statistical screening procedure developed for identifying noisy features involves statistically comparing the saliency of candidate features with the saliency of a known noisy feature. The neural network selection algorithms are developed by posing the neural network model as a nonlinear regression statistical model, and using the likelihood ratio test statistic within a backwards sequential procedure to search for a parsimonious model with equivalent prediction accuracy. Additionally, a comprehensive statistically-based methodology is developed for identifying both a good feature set and an appropriate neural network architecture for a specific situation.

14. SUBJECT TERMS
neural networks, feature saliency, feature selection, model selection, nonlinear regression, likelihood ratio test statistic, hypothesis testing, backwards sequential selection, hidden nodes

15. NUMBER OF PAGES
213

16. PRICE CODE

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
| Unclassified | Unclassified | Unclassified | UL |